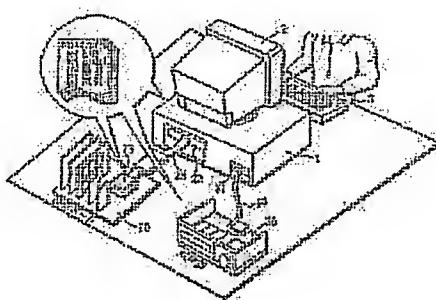


DEBUGGING DEVICES CAPABLE OF TAKING OVER OPERATION FROM EACH OTHER BETWEEN HARDWARE ENVIRONMENTS WHILE RUNNING PROGRAMS THEREIN**Publication number:** JP10326203 (A)**Publication date:** 1998-12-08**Inventor(s):** NISHIHATA MOTOHIDE; IWAMURA YOSHIYUKI; SUMI FUMIO**Applicant(s):** MATSUSHITA ELECTRIC IND CO LTD**Classification:****- international:** G06F11/28; G06F11/22; G06F11/28; G06F11/22; (IPC1-7): G06F11/22; G06F11/28**- European:****Application number:** JP19970163127 19970619**Priority number(s):** JP19970163127 19970619; JP19960157841 19960619; JP19970078420 19970328**Abstract of JP 10326203 (A)**

PROBLEM TO BE SOLVED: To complement advantages and disadvantages of hardware environments having mutually different functions with each other and to enable supervisory control by specifying a switching source and a switching destination and reading operation state information from a switching source operation-verified object, and converting and setting a form in a switching destination operation-verified object. **SOLUTION:** A built-in program developed for a trial machine 30 is run on trial by either of its emulator machine (host computer) or software simulator (host computer 1) to verify the operation. If a specific indication by an operator is detected here, a stored operation-verified object is specified as a switching source operation-verified object and an operation-verified object different from it is specified as a switching destination operation-verified object. The operation state information is read out of the switching source and converted to the format of the switching destination. Then the operation information which is converted in format is set in the switching destination operation-verified object and the operation of the built-in program is restarted at the switching destination operation-verified object.



Data supplied from the **esp@cenet** database — Worldwide

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平10-326203

(43)公開日 平成10年(1998)12月8日

(51)Int.Cl.⁶

G 06 F 11/22
11/28

識別記号

3 4 0
3 4 0

F I

G 06 F 11/22
11/28

3 4 0 A
3 4 0 C

審査請求 未請求 請求項の数17 O L (全 36 頁)

(21)出願番号 特願平9-163127

(22)出願日 平成9年(1997)6月19日

(31)優先権主張番号 特願平8-157841

(32)優先日 平8(1996)6月19日

(33)優先権主張国 日本 (JP)

(31)優先権主張番号 特願平9-78420

(32)優先日 平9(1997)3月28日

(33)優先権主張国 日本 (JP)

(71)出願人 000005821

松下電器産業株式会社
大阪府門真市大学門真1006番地

(72)発明者 西畠 素秀

大阪府門真市大学門真1006番地 松下電器
産業株式会社内

(72)発明者 岩村 喜之

大阪府門真市大学門真1006番地 松下電器
産業株式会社内

(72)発明者 角 史生

大阪府門真市大学門真1006番地 松下電器
産業株式会社内

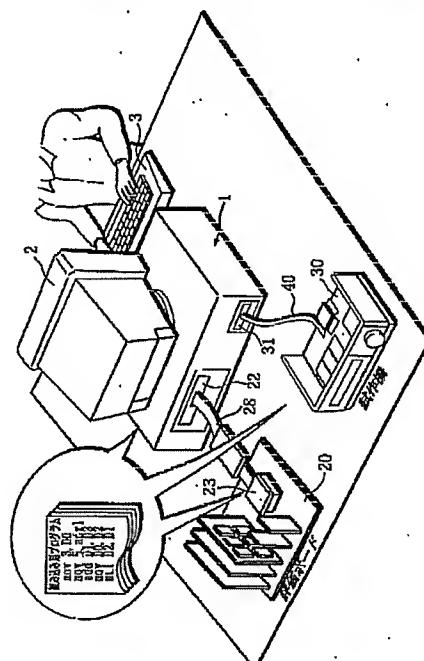
(74)代理人 弁理士 中島 司朗

(54)【発明の名称】複数のハードウェア環境上においてプログラムを別々に動作させつつも、ハードウェア環境間で動作状態を継承し合うことができるデバッグ装置

(57)【要約】

【課題】所定のターゲットマシン向けに開発された組み込みプログラムを、当該ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち、何れかにおいてテスト動作させ、これの動作検証を行う

【解決手段】ハードウェア環境を切り換える旨のコマンドを操作者が入力すると、ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータが有している記憶資源のメモリ内容、レジスタ内容の保持値を他の何れかに転送する。情報を転送後、必要に応じてブレークポイント等のデバッグのための設定情報を他の何れかに再設定し、他のハードウェア環境でプログラムの実行を再開する。これにより現在デバッグ中のハードウェア環境が有する欠点を他のハードウェア環境の優位点でカバーすることもできる。



【特許請求の範囲】

【請求項1】 所定のターゲットマシン向けに開発された組み込みプログラムを当該ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち、何れかにおいてテスト動作させ、これの動作検証を行うデバッグ装置であって、

前記ターゲットマシン、エミュレータマシン、ソフトウェアシミュレータは、組み込みプログラムの動作状態を示す動作状態情報を有し、動作状態情報の入出力を固有の様式にて行い、

ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち何れかを動作検証対象として記憶する動作検証対象記憶手段と、

操作者によるコマンド入力を受け付ける受付手段と、受付手段が受け付けたコマンド内に所定指示が含まれていればこれを検出する検出手段と、

所定指示が検出されると、動作検証対象記憶手段に記憶されている動作検証対象を切換元動作検証対象に指定し、受付手段が受け付けたコマンドに基づいて前記ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち切換元動作検証対象と異なるものを切換先動作検証対象に指定する指定手段と、

切換元及び切換先が指定されると、切換元動作検証対象から動作状態情報を読み出す読み出手段と、

読み出された動作状態情報を、切換先動作検証対象の様式に変換する変換手段と、

動作状態情報が切換先の様式に変換されれば、様式が変換された動作状態情報を切換先動作検証対象に設定する設定手段と、

動作状態情報が設定されると、組み込みプログラムの動作を切換先動作検証対象において再開するよう制御する動作再開手段とを備えることを特徴とするデバッグ装置。

【請求項2】 前記ターゲットマシンは、動作状態情報の入出力を第1の様式にて行い、前記エミュレータマシンは、動作状態情報の入出力を第2の様式にて行い、前記ソフトウェアシミュレータは、動作状態情報の入出力を第3の様式にて行い、

前記変換手段は、

切換元がターゲットマシンであり切換先がソフトウェアシミュレータであれば、第1の様式を有する動作状態情報を第3の様式に変換する第1変換部と、

切換元がエミュレータマシンであり切換先がソフトウェアシミュレータであれば、第2の様式を有する動作状態情報を第3の様式に変換する第2変換部とを備え、

前記設定手段は、

動作状態情報が第3の様式に変換されれば、動作状態情

報をソフトウェアシミュレータに設定する第1設定部を備えることを特徴とする請求項1記載のデバッグ装置。

【請求項3】 前記デバッグ装置において前記変換手段は、

切換元がソフトウェアシミュレータであり切換先がターゲットマシンであれば、第3の様式を有する動作状態情報を第1の様式に変換する第3変換部と、

切換元がソフトウェアシミュレータであり切換先がエミュレータマシンであれば、第3の様式を有する動作状態情報を第2の様式に変換する第4変換部とを備え、

前記設定手段は、

動作状態情報が第1の様式に変換されれば、動作状態情報をターゲットマシンに設定する第2設定部と、

動作状態情報が第2の様式に変換されれば、動作状態情報をエミュレータマシンに設定する第3設定部とを備えることを特徴とする請求項2記載のデバッグ装置。

【請求項4】 前記デバッグ装置において前記変換手段は、

切換元がターゲットマシンであり切換先がエミュレータマシンであれば、第1変換部を制御して、第1の様式を有する動作状態情報を第3の様式に変換させ、

変換後、第4変換部を制御して、第3の様式を有する動作状態情報を第2の様式に変換させる第1制御部と、

切換元がエミュレータマシンであり切換先がターゲットマシンであれば、第2変換部を制御して、第2の様式を有する動作状態情報を第3の様式に変換させ、変換後、第3変換部を制御して、第3の様式を有する動作状態情報を第1の様式に変換させる第2制御部とを備えることを特徴とする請求項3記載のデバッグ装置。

【請求項5】 前記デバッグ装置は、

エミュレータ内部のプロセッサと第1回線により接続され、第1の様式に変換された動作状態情報の入出力をを行うシリアル転送ポートと、

ターゲットマシン内部のインサーキットエミュレータマシンと第2回線により接続され、第2の様式に変換された動作状態情報の入出力をを行うエミュレータマシンドライバ部とを備えることを特徴とする請求項4記載のデバッグ装置。

【請求項6】 前記動作状態情報は、動作検証対象内のレジスタの保持値と、メモリの保持値とからなることを特徴とする請求項5記載のデバッグ装置。

【請求項7】 前記メモリの保持値は、

プロセッサに動作停止を指示するブレークポイント、組み込みプログラムを構成する命令のうち、実行された命令のアドレスからなるトレースデータ、操作者が定めた所定条件を有し、当該成立時に割り込み信号を発生させる割込設定情報の何れかであることを特徴とする請求項6記載のデバッグ装置。

【請求項8】 前記デバッグ装置は更に、ターゲットマシン、エミュレータマシン、ソフトウェア

シミュレータが有するレジスタ及びメモリを、有効値が格納されているものと、無効値が格納されているものとに分類して管理する記憶資源管理手段と、
読み出手段により切換元動作検証対象から動作状態情報が読み出されると、記憶資源管理手段の管理内容を参照して、読み出された動作状態情報に含まれている無効値をデフォルト値に書き換える書換手段とを備え、
前記変換手段は、

無効値がデフォルト値に書き換えられた動作状態情報を、切換先動作検証対象の様式に変換することを特徴とする請求項7記載のデバッグ装置。

【請求項9】 前記デバッグ装置は更に、
ターゲットマシン、エミュレータマシン、ソフトウェアシミュレータが有するメモリにおいて、ブレークポイントの設定が可能な領域と、不可能な領域とに分割して管理する設定可否領域管理手段と、
動作状態情報が変換されると、設定可否領域管理手段を参照して、ブレークポイントが切換先動作検証対象においても同様に設定することができるか否かを判定する判定手段と、
切換先動作検証対象においてブレークポイントが設定できない場合、切換先ではブレークポイントが設定できない旨を操作者に通知する通知手段とを備えることを特徴とする請求項7記載のデバッグ装置。

【請求項10】 前記デバッグ装置は更に、
各動作検証対象が蓄積できるアドレスの最大量を管理する最大量管理手段と、
動作状態情報が変換されると、最大量管理手段におけるアドレスの最大量を参照して切換先動作検証対象における命令アドレス群の蓄積量を認識する認識手段と、
蓄積量を認識すると、切換先動作検証対象のアドレス蓄積量と、切換先動作検証対象とのアドレス蓄積量の違いを操作者に通知する通知手段とを備えることを特徴とする請求項7記載のデバッグ装置。

【請求項11】 前記デバッグ装置は更に、
各動作検証対象が割り込み発生条件の成立、不成立を監視する能力を有するか否かを動作検証対象毎に管理する監視能力管理手段と、
動作状態情報が切換先の様式に変換されると、切換先動作検証対象が当該条件の成立を監視する能力を有するか否かを判定する判定手段と、
能力を有さない場合、切換先動作検証対象には、成立、不成立を監視する能力がない旨を操作者に通知する通知手段とを備えることを特徴とする請求項7記載のデバッグ装置。

【請求項12】 前記デバッグ装置は更に、
受付手段が受け付けたコマンド内に所定指示が含まれていない場合、同コマンドに割込信号発生を監視する旨の指示が含まれていればこれを解読する第1解読手段と、
同コマンドに発生した割込信号の信号番号と、当該番号

に応じた切換先が含まれていればこれを解読する第2解読手段と、
監視する旨の指示が解読されると、現在動作中の切換元の動作検証対象における割込信号発生を監視する監視手段と、
監視中割込信号が発生すると、その割込信号が解読された信号番号に合致するかを判定する判定手段とを備え、
前記読み出手段は、

信号番号と合致すると、切換元動作検証対象から動作状態情報を読み出すことを特徴とする請求項7記載のデバッグ装置。

【請求項13】 前記デバッグ装置は更に、
受付手段が受け付けたコマンドに所定指示が含まれていない場合、同コマンド内に所定メモリ或はレジスタの内容書き換えを監視する旨の指示が含まれていればこれを解読する第1解読手段と、
同コマンドに、内容書き換えが発生した場合に切り換えるべき切換先の指示が含まれていればこれを解読する第2解読手段と、

書き換えを監視する旨の指示が含まれていると、動作状態情報内の所定メモリ或はレジスタの内容の書き換えを監視する監視手段とを備え、
前記読み出手段は、

切換先動作検証対象が規定されれば、動作状態情報を読み出し、
前記変換手段は、

読み出された動作状態情報を、コマンドに含まれていた切換先動作検証対象の様式に変換することを特徴とする請求項7記載のデバッグ装置。

【請求項14】 前記デバッグ装置は更に、
受付手段が受け付けたコマンドに所定指示が含まれていない場合、動作検証対象の切換指示が含まれていればこれを解読する第1解読手段と、
操作者が組み込みプログラム内の複数の領域にどの動作検証対象を割り当てるべきかを示す割当表が同コマンドに含まれていれば、これを解読する第2解読手段と、
切換元動作検証対象における領域間の移動タイミングを検出する旨の指示が同コマンドに含まれていればこれを解読する第3解読手段と、
監視させる旨の指示が解読されると、切換元動作検証対象内の組み込みプログラムにおける複数領域間の移動を監視する監視手段とを備え、
前記指定手段は、

移動の監視中、複数領域間の移動が確認されると、領域に対応づけられた動作検証対象を切換先動作検証対象に規定する切換先規定部とを備え、
前記読み出手段は、
切換先動作検証対象が規定されれば、切換元動作検証対象から動作状態情報を読み出すことを特徴とする請求項7記載のデバッグ装置。

【請求項15】 前記デバッグ装置は更に、受付手段が受け付けたコマンド内に所定指示が含まれていない場合に動作検証対象の切換指示が含まれていればこれを解読する第1解読手段と、前記組み込みプログラム中で使用される変数がとり得る複数の値域に操作者がどの動作検証対象を割り当てたかを示す割当表が同コマンドに含まれていればこれを解読する第2解読手段と、組み込みプログラムにおける所定変数の値の変化を参照する旨の指示が同コマンドに含まれていればこれを解読する第3解読手段と、変化を参照する旨が解読されると、変数に対応するレジスタ或はメモリの保持値が区分された複数値域のうち、何れに該当するかを判定する判定手段とを備え、前記指定手段は、判定された領域に対応づけられた動作検証対象を切換先動作検証対象に規定する切換先規定部を備え、前記読出手段は、切換先動作検証対象が規定されれば、切換元動作検証対象から動作状態情報を読み出すことを特徴とする請求項7記載のデバッグ装置。

【請求項16】 前記デバッグ装置は、受付手段が受け付けたコマンド内に所定指示が含まれていない場合、前記組み込みプログラム中で使用されるパラメータがとり得る複数の値域に操作者がどの動作検証対象を割り当てたかを示す割当表が同コマンドに含まれていればこれを解読する第1解読手段と、組み込みプログラムにおける所定パラメータの値を参照する旨の指示が同コマンド内に含まれていればこれを解読する第2解読手段と、現在のパラメータの値の変化が検出されると、変化後の値が区分された複数領域のうち、何れに該当するかを判定する判定手段とを備え、前記指定手段は、判定された値域に対応づけられた動作検証対象を切換先動作検証対象に規定する切換先規定部を備え、前記読出手段は、切換先動作検証対象が規定されれば、切換元動作検証対象から動作状態情報を読み出すことを特徴とする請求項7記載のデバッグ装置。

【請求項17】 前記デバッグ装置は、受付手段が受け付けたコマンド内に所定指示が含まれていない場合、操作者が複数の動作検証対象にどの実行予定時刻を対応付けたかを示す対応表が同コマンド内に含まれていればこれを解読する第1解読手段と、現在時刻を計時する計時手段とを備え、前記指定手段は、計時されている現在時刻が実行予定時刻に到達すると、その動作検証対象を切り替え先に規定する切換先規定部を備え、

前記読出手段は、切換先動作検証対象が規定されれば、切換元動作検証対象から動作状態情報を読み出すことを特徴とする請求項7記載のデバッグ装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、組み込み用途プログラムのデバッグを行うデバッグ装置に関する。

【0002】

【従来の技術】今日の家電製品にとってマイコンシステムは必要不可欠な存在である。家電製品向けのプログラムの開発は、大量生産に耐え得る品質が要求されるので、デバッグ作業の徹底が求められる。ここでいうデバッグ作業とは プログラムをハードウェア環境上で動作させ、その動作時において現れた不具合(バグ)の原因を追究してこれを修正する作業をいう。

【0003】プログラム(デバッグされるプログラムであるから、被デバッグプログラムという)の動作形態には、以下の①～③のものがある

- ①被デバッグプログラムを家電製品の試作機上で動作させるもの。
- ②被デバッグプログラムを評価ボード上で動作させるもの。
- ③被デバッグプログラムをホストコンピュータ上のシミュレータを用いて模擬動作させるもの。

【0004】①のようなハードウェア環境の形態は、プログラムのモニタリングを行なうことを主目的にしたものである。プログラムのモニタリングを主目的としたハードウェア環境は『モニタ型』という略称で称呼される。モニタ型のハードウェア環境は、ターゲットのマイクロコンピュータやマイクロプロセッサの試作機が完成した際、その動作確認や簡単なデバッグ作業にもっぱら使用される。

【0005】②のようなハードウェア環境の形態は、インサーキットエミュレータと、様々なデバッグ用ツールとを用いた高度なデバッグ作業を可能としている。このようなハードウェア環境は『ICE(In-Circuit Emulator)型』という略称で称呼される。③のようなハードウェア環境の形態は、シミュレータ型と呼ばれる。シミュレータには、命令レベルのもの、論理レベルのものの二種類がある。命令レベルソフトウェアシミュレータとは、マイクロコンピュータやマイクロプロセッサの命令をホストのオペレーティングシステム上のソフトウェアでシミュレーション実行させることによりマイクロコンピュータやマイクロプロセッサと同じ実行状態を模擬的に他ホスト上に作りだすソフトウェアシステムである。

【0006】論理レベルソフトウェアシミュレータとは、通常マイクロコンピュータやマイクロプロセッサのチップの設計に使用され、ハードウェアと同じ動作を他ホスト上にソフトウェアでシミュレーション実行するも

のである。

【0007】

【発明が解決しようとする課題】ところで上記の3つの形態のハードウェア環境には、動作検証能力に限界がある。モニタ型ハードウェア環境の動作検証能力の限界とは、モニタリングを前提にしたハードウェア環境であるから、動作検証中にバグが発見されても、その原因を解析する能力が不充分であり、バグ発生に思うように対処できない点である。ICE型ハードウェア環境では、トレース機能を用いてプログラム実行の履歴を確認する等の対処が可能であるが、モニタ型ハードウェア環境では、メモリを多く用いるトレース機能等は利用できない場合が多く、操作者は、貧弱な機能を駆使して原因を追求せねばならない。また、ターゲットマシンの使用時の様々なケースを想定しての動作確認が不充分となる点も軽視できない。『ターゲットマシンの使用時の様々なケース』の代表的なものには、プログラムの多重割り込み発生時や、指定時刻での割り込み発生時の動作確認等、割込み発生に関連するものが多い。これらをターゲットマシン上で実現するには、ハードウェアの一部改造等が必要となるが、現実問題としてこのような改造はそうた易く行えるものではなく、ターゲットマシンで動作確認できる内容は限られてしまう。

【0008】ICE型ハードウェア環境の動作検証能力の限界とは、所詮ターゲットマシンとは回路構成が違う点である。即ち、評価ボード上でたとえ完全に動作するに見えて、その動作により出荷OKか否かの最終判断を下すことはできない。またターゲットマシンの正規の形態ではないため、周辺機器をも含めた動作検証が不充分な場合がある。何故なら評価ボードにおける配線長は、試作機実物のものよりも長い場合があり、この長さ故に、周辺機器の制御にタイムラグが生じていることも多々ある。このようなタイムラグのため、ICE型ハードウェア環境における周辺機器をも含めた動作検証は厳密に欠けてしまう。

【0009】ソフトウェアシミュレータ型ハードウェア環境の限界とは、シミュレータ自体がホストコンピュータを利用して動作しているため、プログラムの動作がモニタ型またはICE型よりもかなり鈍くなるという点である。例えば、ICE型やモニタ型のハードウェア環境において1秒で処理が終了するものが、数分から数十分かかる。

【0010】個々ハードウェア環境の動作検証能力の限界のため、バグ原因作業の究明が捗らない場合、開発者はバグが発生した当時の状況を別のハードウェア環境において再現したいと考える。しかし他のハードウェア環境で再現するには、バグ発生当時のレジスタ、メモリの保持値を他のハードウェア環境上に設定したり、マイクロコンピュータやマイクロプロセッサのプログラムを最初から実行し直す等の試行錯誤を繰り返す必要がある。

このようにバグ発生当時の状況を再現するために試行錯誤を繰り返せば、多くの開発時間が無駄に費やされてしまう。

【0011】本発明の目的は、それぞれが異なる機能を有するハードウェア環境の長所、短所を互いに補完しあえるように、統括して制御することができるデバッグ装置を提供することである。本発明の別の目的は、開発者の手間を煩わせることなく、バグ発生時のハードウェア環境により近い状況を他のハードウェア環境上で再現することができるデバッグ装置を提供することである。

【0012】

【課題を解決するための手段】上記の目的を達成するデバッグ装置は、所定のターゲットマシン向けに開発された組み込みプログラムを当該ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち、何れかにおいてテスト動作させ、これの動作検証を行うデバッグ装置であって、前記ターゲットマシン、エミュレータマシン、ソフトウェアシミュレータは、組み込みプログラムの動作状態を示す動作状態情報を有し、動作状態情報の入出力を固有の様式にて行い、ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち何れかを動作検証対象として記憶する動作検証対象記憶手段と、操作者によるコマンド入力を受け付ける受付手段と、受付手段が受け付けたコマンド内に所定指示が含まれていればこれを検出する検出手段と、所定指示が検出されると、動作検証対象記憶手段に記憶されている動作検証対象を切換元動作検証対象に指定し、受付手段が受け付けたコマンドに基づいて前記ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち切換元動作検証対象と異なるものを切換先動作検証対象に指定する指定手段と、切換元及び切換先が指定されると、切換元動作検証対象から動作状態情報を読み出す読出手段と、読み出された動作状態情報を、切換先動作検証対象の様式に変換する変換手段と、動作状態情報が切換先の様式に変換されれば、様式が変換された動作状態情報を切換先動作検証対象に設定する設定手段と、動作状態情報が設定されると、組み込みプログラムの動作を切換先動作検証対象において再開するよう制御する動作再開手段とを備えることを特徴としている。

【0013】

【発明の実施の形態】以降デバッグ装置の実施形態について説明する。本デバッグ装置のハードウェア構成は汎用コンピュータにICE制御ボード22、シリアル転送ポート31を接続して構成される。図1は、本デバッグ装置の外観図である。家電機器の試作機30はモニタ型のハードウェア環境を構築する。モニタ型であるから、組み込み用途のプログラムが格納されたROMを有し、当

該被デバッグプログラムを試作機30上で動作させることができる。試作機30における被デバッグプログラムはホストコンピュータ1の管轄下にあり、被デバッグプログラムの動作開始、動作終了は、シリアル回線40を介したホストコンピュータ1からの指示があつて初めて行われる。ホストコンピュータ1は試作機30上のメモリ、レジスタ等記憶資源のアクセス権も有し、所望のタイミングで、試作機30の記憶資源の内容を記憶資源の状態情報として読み出すことができる。

【0014】評価ボード20は、被デバッグプログラムを動作させることができる。評価ボード20における被デバッグプログラムもホストコンピュータ1の管轄下にあり、被デバッグプログラムの動作開始、動作終了は、パラレル回線28を介したホストコンピュータ1からの指示に従つて行う。ホストコンピュータ1は評価ボード20上のメモリ、レジスタ等記憶資源のアクセス権も有し、所望のタイミングで、評価ボード20の記憶資源の内容を記憶資源の状態情報として読み出すことができる。

【0015】図2は、デバッグ装置のハードウェア構成図である。図2に示すように、ホストコンピュータ1はCPU、ディスプレイ、メモリ、ハードディスク、キーボードを有する汎用コンピュータであり、ICE制御ボード22、シリアル転送ポート31が接続されている。ICE制御ボード22及びシリアル転送ポート31はそれぞれ評価ボード20、試作機30と接続されている。

【0016】試作機30上のマイコンシステムのハードウェア構成を図3に示す。図3において試作機30は、プロセッサ33、メモリ34、周辺I/Oレジスタ群35が実装されている。プロセッサ33は、被デバッグプログラムを構成する機械語命令を逐一読み出す命令読み出し回路と、読み出された機械語命令を解読する解読器と、解読結果が演算命令であると解読結果が下された場合にその特定演算命令の演算を行う演算器と、レジスタ群と、命令読み出し回路に命令の読み出し先を指示するプログラムカウンタとを有する。プロセッサ33は専用端子39を有し、この専用端子39に所定の指示信号が送出されると、プロセッサ33は試作機30の動作状態を示す動作状態情報を読み出し、専用端子39を介してホストコンピュータ1に出力する。

【0017】メモリ34には、ユーザ空間にモニタリングプログラム36と、被デバッグプログラム37と、被デバッグプログラム37が試作機30内部で作業用に利用するターゲット内ワークエリア38とが配されている。被デバッグプログラム37は、デバッグの対象となる組み込み用途のプログラムであり、試作機30が起動されるとプロセッサ33により実行される。被デバッグプログラム37はモニタリングの対象となるため、モニタリング命令が組み込まれている。モニタリング命令とは、プロセッサ33が専用端子39を介してデータを受

け取ったとき、被デバッグプログラム37からモニタリングプログラム36へと分岐する旨をプロセッサ33に指示する命令である。

【0018】モニタリングプログラム36は、専用端子39からのデータの入力待ちをプロセッサ33に行わせる。もし専用端子39にデータが入力されると、そのデータに従つた動作をプロセッサ33に行わせる。周辺I/Oレジスタ[i] (i=1,2,3 n) 35は、周辺I/Oの値を保持するレジスタである。ここで周辺I/Oとは、シリアルインターフェース、パラレルインターフェース、割り込みコントローラ、A/D変換器、D/A変換器、タイマカウンタなどであり、これらの値が周辺I/Oレジスタに個別に格納されていることにより、プロセッサ33は任意の周辺I/Oの値を取り出すことができる。

【0019】評価ボード20上のマイコンシステムのハードウェア構成を図4に示す。図4において評価ボード20は、プロセッサ33に代えてICEチップ23が取り付けられ、メモリ24、周辺I/Oレジスタ群25を有する。メモリ24は、スーパーバイザ空間と、ユーザ空間とを有する。スーパーバイザ空間にはスーパーバイザプログラム26及びデバッグ情報エリア27が配されており、ユーザ空間には被デバッグプログラム37、ターゲット内ワークエリア38が配されている。

【0020】周辺I/Oレジスタ群25は、周辺I/Oであつて、評価ボード20上で動作可能なもの保持値を保持するレジスタである。スーパーバイザプログラム26は、ブレークポイントを設定するブレークポイント設定ツールと、ICEチップ23が命令を実行する度に、実行された命令のアドレスを蓄積するトレースツールと、操作者が定めた所定条件の成立、不成立を監視し、当該成立時に割り込み信号を発生させる割り込み設定ツールとを有する。

【0021】デバッグ情報エリア27は、スーパーバイザプログラム26がワークエリアとして用いているメモリ領域であり、前記ブレークポイントの設定先アドレスと、前記ブレークポイントの動作の有効、無効に関するブレークポイント情報と、蓄積された複数の命令アドレスからなるトレースデータと、操作者が定めた割り込み信号発生のための所定条件を示す割込信号発生要件情報とが格納されている。

【0022】図27(a)は、ブレークポイント設定先アドレスの一例を示す図である。図中の『BREAK_POINT[1]=0x0020』は一個目のブレークポイントを0x0020から始まる所定のサイズの領域に設定する旨を示し、『BREAK_POINT[2]=0x0250』は二個目のブレークポイントを0x0250から始まる所定のサイズの領域に設定する旨を示している。『BREAK_POINT[3]=0x0300』は、三個目のブレークポイントを0x0300から始まる所定のサイズの領域に設定する旨を示している。これらのブレークポイント設定先アドレスは、これまでのスーパーバイザプログラム2

6による被デバッグプログラム37のデバッグ時に、操作者がコマンドを用いてブレークポイントを設定し、プロセッサ33がスーパバイザプログラム26内のブレークポイント設定ツールを実行したためにメモリ24に書き込まれている。

【0023】図27(b)は、割込信号発生要件の一例を示す図である。図中の『CONDITION[1]=20cycle, Int3』は、命令実行が実行時間の単位長の20倍だけ繰り返された後に、3番の割り込み信号を発生させる旨を示し、『CONDITION[2]=150cycle, Int5』命令実行が実行時間の単位長の150倍だけ繰り返された後に、5番の割り込み信号を発生させる旨を示している。これらの割込信号発生要件は、これまでの被デバッグプログラム37のデバッグ時に、操作者が割り込み発生設定操作を行い、被デバッグプログラム37がスーパバイザプログラム26内の割り込み信号発生ツールを実行したためにメモリ24に書き込まれている。

【0024】図27(c)は、トレースデータの一例を示す図である。図中の『0x0000,0x0001,0x0002,0x0003,0x0004,0x0005』は、ICEチップ23が、被デバッグプログラム37内の0アドレスx0000の機械語命令、アドレス0x0001の機械語命令、アドレス0x0002の機械語命令、アドレス0x0003の機械語命令を既に実行していることを示している。これらのトレースデータは、これまでのICE型ハードウェア環境による被デバッグプログラム37のデバッグ時に、操作者がTRACEコマンドを入力し、プロセッサ33がスーパバイザプログラム26内のトレースツールを実行したためにメモリ24に書き込まれている。

【0025】ターゲット内ワークエリア38は被デバッグプログラムが用いるワークエリアである。図6は、図1に示したホストコンピュータ1の内部階層図である。ホストコンピュータ1の内部階層は、アプリケーション層11、オペレーティングシステム層12、BIOS層13、物理層14からなり汎用的なコンピュータの階層構成と何等変わりはない。このような汎用的な階層構成においてアプリケーション層11にはシェミレータ10が存在する。シェミレータ10がアプリケーション層11上に存在するのは、シェミレータ10はオペレーティングシステム上で起動するアプリケーションプログラムの一種であることを意味する。ICE制御ボード22及びシリアル転送ポート31は物理層14に位置することがわかる。ICE制御ボード22については専用のデバイスドライバがBIOS層13に存在する(ICE用デバイスドライバ21参照)。

【0026】シェミレータ10がアプリケーション層11上に存在するから、シェミレータ型ハードウェア環境の窓口は、アプリケーション層11となる。デバイスドライバ21がBIOS層13上に存在するから、ICE型ハードウェア環境の窓口は、BIOS層13となる。シリアル転送ポート31が物理層14上に存在するから、モニタ型

ハードウェア環境の窓口は、物理層14となる。

【0027】窓口が互いに異なる階層に存在するため、各ハードウェア環境にパラメータを受け渡す際には、パラメータを各階層に適した様式に変更せねばならない。またハードウェア環境同士でパラメータの受け渡す際には、様式の変換が必要となる。具体的にはシェミレータ10はアプリケーション層11上に存在するアプリケーションプログラムの一種であるから引数様式によりパラメータを受け渡しする必要がある。デバイスドライバ21はBIOS層13に存在するデバイスドライバであるから、BIOSコマンド様式によりパラメータを受け渡しする必要がある。シリアル転送ポート31は物理層14に存在するから、I/Oポートを用いてパラメータを受け渡しする必要がある。

【0028】ICE型、モニタ型、シェミレータ型という3つのハードウェア環境を管理するため、本装置は、ICE型、モニタ型、シェミレータ型にそれぞれ、TARGET1, TARGET2, TARGET3という識別子を付しており、また図示しない原簿テーブルに現状デバッグ対象に設定されているハードウェア環境の識別子を登録している。図7は、アプリケーション層11の内部においてICE型、モニタ型、シェミレータ10型のハードウェア環境の統合のために形成された階層を示す図である。

【0029】ターゲット依存層53は、ICE型、モニタ型、シェミレータ型のハードウェア環境間が用いているデータ様式を引数様式に統一するため階層である。ターゲット相互接続層52は、共通様式に統一されたデータをICE型、モニタ型、シェミレータ10型が共有し合うための階層であり、共有用のワークエリアを含む。

【0030】デバッグ核層51は、いわゆるシンボルデバッガのアプリケーション部に相当する機能を有する階層である。ユーザインターフェイス層50は、オペレーティングシステム層12の標準入出力ライブラリを用いてユーザインターフェイスを操作者に提供する階層である。即ち、ユーザインターフェイス層50は、オペレーティングシステム層12の標準入出力ライブラリを利用して、操作者からの指示を仰ぐメッセージをディスプレイ2に表示する。コマンドが入力されると、そのコマンドの解釈及び実行をデバッグ核層51以下の階層に委ねる。デバッグ核層51以下の階層が入力されたコマンドを実行すると、ユーザインターフェイス層50はコマンドの実行結果をオペレーティングシステム層12の標準入出力ライブラリを利用してディスプレイ2に表示する。

【0031】図8は、デバッグ核層51、ターゲット相互接続層52、ターゲット依存層53のそれぞれの階層にどのようなモジュールが存在するかを示した図である。本図においてデバッグ核層51には、シンボルアドレスマップ71、コマンドインタプリタ72、及びコマンド変換ルーチン73が存在することがわかる。シン

ボルーアドレスマップ71は、シンボルと、アドレスとの対応関係を示す情報である。ここでシンボルとは、ソースプログラム内で使用されている変数名、ライブラリ等をいい、シンボルーアドレスマップ71は、これらシンボルが示す変数名がメモリ領域のどのアドレスに対応しているかを示す。具体的には、シンボルーアドレスマップ71はメモリサイズ、変数の型、ソースコードの一行が存在する行の行番号と、その行に対応する実行コードのアドレスとを対応づけるライン情報などから構成される。

【0032】コマンドインタプリタ72は、ユーザインターフェイス層50に対してプログラム開発者が入力したコマンドを解析し、その解析内容に従った指示をコマンド変換ルーチン73に通知する。ここでコマンドインタプリタ72は、キータイプ入力されたコマンドが図9に示す体系図内の何れかに該当するものとして解析を行う。

【0033】図9に記載されているコマンドのうち、runコマンド、stopコマンド、stepコマンド、setBPコマンド、clrBPコマンド、SetINTコマンド、readMEMコマンド、readREGコマンド、writeMEMコマンド、writeREGコマンド、TRACEコマンドは一般的なデバッグ装置向けのコマンドである。runコマンドは、コマンド発行先に指定されているハードウェア環境に、被デバッグプログラムの動作開始を指示するコマンドであり、そのオペランドにおいて被デバッグプログラムの動作開始アドレスを指定させることができる。

【0034】stopコマンドは、コマンド発行先に指定されているハードウェア環境に、被デバッグプログラムの動作停止を指示するコマンドである。stepコマンドは、コマンド発行先に指定されているハードウェア環境に、被デバッグプログラムの命令を一命令ずつ実行する旨を指示するコマンドであり、そのオペランドにおいて被デバッグプログラムの動作開始アドレスを指定させることができる。

【0035】setBPコマンドは、コマンド発行先に指定されているハードウェア環境に、オペランドに指定されている被デバッグプログラム内のアドレスへとブレークポイントを設定させる旨を指示するコマンドである。ここでいう『ブレークポイント設定』とは、被デバッグプログラム内に所定の割り込み信号を発生させる命令を書き込むこと、或は、動作停止すべき被デバッグプログラム内のアドレスをプロセッサ内の所定のレジスタに登録することをいうが、本実施形態では、これらの動作と共に、割り込み信号発生命令の書き込み先アドレスもハードウェア環境内のメモリに書き込まれる。

【0036】clrBPコマンドは、オペランド『breakNO』で指示される番号のブレークポイントを発生する旨の割り込み信号発生命令及び当該命令の書き込み先アドレスを削除するコマンドである。SetINTコマンドは、オペラ

ンド『CycleNumber』で指定された実行時間（この実行時間は、実行サイクルの整数倍数で表現されることは留意されたい。）だけ命令実行が繰り返されたかをハードウェア環境に監視させ、もし当該実行時間だけ繰り返されれば、オペランド『intNO.』で指示される番号の割込信号を発生する旨をハードウェア環境に指示するコマンドである。

【0037】readMEMコマンドは、『TARGET』で指示されたハードウェア環境内のオペランド『s_address』『e_address』で指示された領域の内容を読み出す旨をハードウェア環境に指示する。readREGコマンドは、オペランド『register』で指示された番号のレジスタの内容を読み出す旨をハードウェア環境に指示する。

【0038】writeMEMコマンドは、『TARGET』で指示されたハードウェア環境内のオペランド『address』で指示されたメモリ領域に『value1』で指示された内容を書き込む旨をハードウェア環境に指示する。writeREGコマンドはオペランド『register』で指示された番号のレジスタに『value2』で指示される内容を書き込む旨をハードウェア環境に指示する。

【0039】尚readMEMコマンド、readREGコマンド、writeMEMコマンド、writeREGコマンドはオペランドを変数で指示することも可能である。変数で指示されている場合は、コマンドインタプリタ72はシンボルーアドレスマップ71に登録された情報を参照する。このような情報を参照することにより、変数が存在しているメモリ上のアドレスまたはレジスタ種別とサイズの特定ができる。変数情報取得後、変数の参照コマンドはコマンドインタプリタ72により、あるアドレスからサイズ分の情報をメモリから取得またはあるレジスタの値を取得／設定するというコマンドに置き換えられ、ターゲット相互接続層52を介してシュミレータ10、モニタ制御層61、ICE制御層62の何れかに通知される。

【0040】TRACEコマンドは、ハードウェア環境において実行された命令のアドレスを蓄積する旨をハードウェア環境に指示する。selectTARGETコマンドは、これまでコマンド発行先に指定されていたハードウェア環境の記憶資源の動作状態を他のハードウェア環境に転送する指示と、デバッグコマンドの発行先を他のハードウェア環境に切り換える旨の指示とを含むコマンドである。記憶資源の動作状態を他のハードウェア環境に転送するには、ハードウェア環境のターゲット内ワークエリア38の内容、レジスタの内容、ブレークポイント、割込信号発生要件情報、周辺I/Oレジスタの内容、トレースデータを個別に転送せねばならない。個別転送を実現するため、コマンドインタプリタ72は、サブコマンドcopyMEMコマンド、copyREGコマンド、copyBPコマンド、copyINTコマンド、copyI/Oコマンド、copyTRACE_DATAコマンドを発行する。

【0041】copyMEMコマンドは、現在デバッグに用い

られているハードウェア環境内のメモリ内容のうち、ターゲット内ワークエリア38の内容をselectTARGETコマンドにより指定された切り換え先へと転送するようターゲット相互接続層52に指示する。またcopyMEMコマンドは、オペランドに転送範囲の開始アドレス～終了アドレスを記載することにより任意の領域の転送を指定することが可能であり、また全メモリ空間の転送を指定するスイッチが設定されることにより、全メモリの値を転送するよう指定することもできる。

【0042】copyRGコマンドは、現在デバッグに用いられているハードウェア環境内のレジスタ群の値をselectTARGETコマンドにより指定された切り換え先へと転送する旨を指示するコマンドである。copyBPコマンドは、現在デバッグに用いられているハードウェア環境内のメモリ内容のうち、ブレークポイントの設定先アドレスをselectTARGETコマンドにより指定された切り換え先へと転送する旨を指示するコマンドである。

【0043】copyINTコマンドは、現在デバッグに用いられているハードウェア環境内のメモリ内容のうち、割り込み信号の割込信号発生要件を示す情報をselectTARGETコマンドにより指定された切り換え先へと転送する旨を指示するコマンドである。copyI/Oコマンドは、現在デバッグに用いられているハードウェア環境内の全ての周辺I/Oレジスタを、selectTARGETコマンドにより指定された切り換え先へと転送する旨を指示するコマンドである。

【0044】copyTRACE_DATAコマンドは、現在デバッグに用いられているハードウェア環境内のメモリ内容のうち、トレースデータをselectTARGETコマンドにより指定された切り換え先への転送をターゲット相互接続層52に指示する。デバッグ核層51の構成要素の説明を引き続き行う。コマンド変換ルーチン73は、コマンドインタプリタ72が解釈したコマンド及びコマンドインタプリタ72が自動発行したコマンドをより下位レベルに変換する。ここでいう下位レベルへの変換は、コマンドを関数呼出コードに変換することをいう。

【0045】コマンドインタプリタ72が入力コマンドをreadMEMコマンド等、状態情報の読み出し命令であると解釈した際、関数呼出コードTCI_get_mem(adr, length, unit, image)等の関数呼出コードに変換する。『TCI_get_mem』の『TCI』とは、『Target Core Interface』の略であり、本関数呼出コードがデバッグ核層51とターゲット相互接続層52との間で使用されることを意味する。即ち、本関数呼出コードは、デバッグ核層51～ターゲット相互接続層52間という局所的な区間でのみ用いられる関数呼出コードなのである。『TCI_get_mem』の『get_mem』とは、『メモリの保持値を取得する』という意味である。『adr, length, unit, image』はそれぞれが引数である。adrが開始アドレスを示し、lengthが取得長を示す。unitは、バイト、ワード、ダブルワード

等取得単位を示し、imageはメモリ内容の格納先を示す。

【0046】これらの意味から判るのは、コマンドインタプリタ72が入力コマンドをreadMEMコマンドと解釈すると、デバッグ核層51からターゲット相互接続層52へと関数呼出コードを用いた指示がなされるということである。読み出し先が試作機30、評価ボード20、シミュレータ10のうちどのハードウェア環境であるかを明示していないため、抽象的なレベルにより関数呼出が規定されていることがわかる。

【0047】コマンドインタプリタ72が入力コマンドをwriteREGコマンド等、状態情報の書き込み命令であると解釈した際、関数呼出コードTCI_set_REG(regs, mask)等の関数呼出コードに変換する。『TCI_set_REG』の『set_REG』とは、『レジスタの既値を設定する』という意味である。『regs, mask』はそれぞれが引数である。regsとは、共用ワークエリアにおいて、汎用レジスタ、プログラムカウンタ、状態レジスタ、データレジスタ、アドレスレジスタ等の各種レジスタの値を個別に格納する配列変数の配列名であり、maskとは、引数regsで指示される配列変数の要素（レジスタ保持値）のうち、どれをハードウェア環境内のプロセッサに設定し、どれをハードウェア環境に設定しないかを示すマスク用のビットパターンである。

【0048】同様にコマンドインタプリタ72が、操作者が入力したコマンドを『run』『stop』『step』『set BP』『clrBP』『SetINT』『readREG』『writeMEM』『TRACE』と解釈すると、コマンド変換ルーチン73は『TCI_run』『TCI_stop』『TCI_step』『TCI_setBP』『TCI_clrBP』『TCI_SetINT』『TCI_read_REG』『TCI_write_ME M』『TCI_TRACE』という関数呼出コードに変換してターゲット相互接続層52に伝える。

【0049】コマンドインタプリタ72が入力コマンドをselectTARGETコマンドと解釈した際、自動発行されるcopyMEMコマンド～copyTRACE_DATAコマンドがどのような関数呼出コードに変換されるかを説明する。copyMEMコマンド～copyTRACE_DATAコマンドによる状態情報の転送は二段階の関数呼出コードにより行われる。第1段階とは、『TCI_get～』という関数呼出コードにより切換元に対して記憶資源の状態情報を読み出しを指示することであり、第2段階とは、『TCI_set～』という関数呼出コードにより切換先に対して記憶資源の状態情報の書き込みを指示することである。

【0050】図13は、変換前のcopyMEMコマンド～copyTRACE_DATAコマンドと、変換後の関数呼出コードTCI_get_mem(adr, length, unit, image)～TCI_set_TD(adr, length, unit, image)の対応関係を示す図である。本図を参照すると、サブコマンドCopyMEMは、TCI_get_mem(adr, length, unit, image)という関数呼出コードと、TCI_set_mem(adr, length, unit, image)という関数呼出コードとに変

換されることがわかる。

【0051】関数呼出コードTCI_get_mem(adr,length,unit,image)は、切換元の記憶資源のうちターゲット内ワークエリア38の内容を切換元から読み出す旨を指示し、関数呼出コードTCI_set_mem(adr,length,unit,image)は、切換先の記憶資源のうちターゲット内ワークエリア38に状態情報を設定する旨を指示する。図13において留意すべきは、関数呼出コードの引数の個数である。

【0052】『CopyReg』から変換された関数呼出コード『TCI_get_reg(regs,mask)』、『TCI_set_reg(regs,mask)』のみ、引数が2つであり、それ以外の関数呼出コードは、(adr,length,unit,image)のように引数を4つもっている。これは、内容の違いはあるが、『CopyMEM』『CopyBP』『CopyINT』『CopyI/O』が転送を命じているコマンドは、何れも、記憶資源のうち、メモリに関するものであり（周辺I/Oレジスタ群25、周辺I/Oレジスタ群35の内容はメモリを介したアクセスが可能である。）、(adr,length,unit,image)という引数で、アクセス内容を表現できるからである。

【0053】『CopyBP』における引数について説明する。ブレークポイントの形態は各ハードウェア環境で異なる場合があるが、大多数なものは、被デバッグプログラム内の機械語命令のアドレスを個々に指示するタイプである。命令アドレスを個々に指示する場合は、引数(adr,length,unit,image)のうち『length,unit』は不用である。しかしハードウェア環境の中には、ブレークポイントを領域で指定できる、いわゆるエリアブレークを利用可能なものも存在する。このようなエリアブレークが利用可能なハードウェア環境の接続を前提にしているため、本装置は、引数(adr,length,unit,image)により、ブレークポイントのサイズと、アクセス長の指定を関数呼出コードの引数内に取り入れている。

【0054】図13における『CopyINT』の引数(cycle,intkind)について説明する。引数cycleは、命令実行の実行時間の単位長の整数倍値を示し、intkindは、発生すべき割り込み信号の信号番号を示す。図8においてターゲット相互接続層52は、様式変換ルーチン74、ワークエリアコンテンツ解析モジュール75、共用ワークエリア77からなる。ここでは、様式変換ルーチン74及び、共用ワークエリア77について説明し、ワークエリアコンテンツ解析モジュール75については後述する。

【0055】様式変換ルーチン74は、コマンド変換ルーチン73により変換された関数呼出コードをハードウェア環境に依存した様式に変換する。ここでいう『ハードウェア環境に依存した様式』とは、関数呼出コードが含んでいる呼出先関数名をハードウェア環境を明示した関数名に書き換えることをいう。ここでサブコマンドCopyMemをコマンド変換ルーチン73がTCI_get_mem(adr,l

ength,unit,image)、TCI_set_mem(adr,length,unit,image)という関数呼出コードに変換したものとする。この場合様式変換ルーチン74はこれらの関数呼出コードに含まれている関数名の書き換えを行う。

【0056】先ず様式変換ルーチン74は、TCI_get_mem(adr,length,unit,image)における『TCI_get_mem』という関数名を切換元を明示した関数名に書き換える。例えば、切換元がモニタ型であり、切換先がシミュレータ型である場合に、関数呼出コードTCI_get_mem(adr,length,unit,image)が生成されたものとする。切換元のハードウェア環境はモニタ型であるから、当該関数呼出コードはモニタ型を明示した関数名『MON_get_mem』に書き換えられ、MON_get_mem(adr,length,unit,image)を得る。

【0057】続いて様式変換ルーチン74は、TCI_set_mem(adr,length,unit,image)における『TCI_set_mem』という関数名を切換先を明示した関数名に書き換える。切換先のハードウェア環境はシミュレータ型であるから、シミュレータ型を明示した関数名『SIM_set_mem』に書き換えられる。書き換え結果は、SIM_set_mem(adr,length,unit,image)となる。関数名を書き換えた後、様式変換ルーチン74は書き換え後の関数呼出コードを用いて関数呼出を行う。

【0058】様式変換ルーチン74は同様の書き換えを、図13に示した全ての関数呼出コードについて行う。図14は図13に示した全ての関数呼出コードがどのように書き換えられたかを示す図である。関数名の書き換えを行う際、様式変換ルーチン74は、切換元及び切換先がICE型、モニタ型、シミュレータ型の何れであるかを識別する。切換元は、現状デバッグ対象に設定されているハードウェア環境であるから、原簿テーブルを参照して、本ファイルに登録されている識別子を切換元のハードウェア環境と識別する。切換先は、selectTARGETコマンドのオペランドに設定されているハードウェア環境であるから、様式変換ルーチン74はコマンドインタプリタ72からselectTARGETコマンドを受け取り、selectTARGETコマンドのオペランドに示されている識別子を切換先ハードウェア環境を示すものと解釈する。このようにして切換元、切換先が決定されると、関数呼出コードの関数名の書き換えを行う。

【0059】共用ワークエリア77は、引数Imageにて指定された領域Image等、モニタ型、ICE型、シミュレータ型のハードウェア環境により共用されるワークエリアである。ターゲット依存層53は、シミュレータ10、モニタ制御層61、ICE制御層62からなる。

【0060】シミュレータ10は、図5の構成を有するアプリケーションプログラムである。図5に示すように、シミュレータ10は、プロセッサ模擬ルーチン95、メモリ模擬ルーチン96、I/Oレジスタ模擬ルーチン97から構成される。プロセッサ模擬ルーチン95

は、プロセッサ33の機能を模擬する。メモリ模擬ルーチン96は、メモリ24同様、スーパバイザプログラム26、デバッグ情報エリア27、被デバッグプログラム37、ターゲット内ワークエリア38を記憶する。これらを記憶しているため、シュミレータ型ハードウェア環境のバグ解析能力は、ICE型のそれとほぼ対等といえる。

【0061】I/Oレジスタ模擬ルーチン97は、周辺I/Oレジスタ群35の機能を模擬する。プロセッサ模擬ルーチン95、メモリ模擬ルーチン96、I/Oレジスタ模擬ルーチン97はシュミレータ型を指定した関数呼出コードにより起動する。関数呼出コードSIM_get_mem(adr,length,unit,image)を用いて様式変換ルーチン74により呼び出されれば、プロセッサ模擬ルーチン95はメモリ模擬ルーチン96において開始アドレスadrに格納されているデータを、取得長lengthに指定されているサイズだけ、取得単位unitにて指定されている単位毎に読み出す。読み出されると、プロセッサ模擬ルーチン95はデータを共用ワークエリア77において引数imageに指定されている領域に格納する。

【0062】関数呼出コードSIM_set_mem(adr,length,unit,image)を用いて様式変換ルーチン74により呼び出されれば、プロセッサ模擬ルーチン95は共用ワークエリア77において引数imageに指定されている領域のデータをメモリ模擬ルーチン96内の開始アドレスadr以後の領域に書き込む。書き込み作業は引数lengthに指定されているサイズだけ、引数unitにて指定されている單位で行われる。

【0063】モニタ制御層61は、試作機30における被デバッグプログラムの管轄権及び試作機30上のメモリ、レジスタ等記憶資源のアクセス権も有し、所望のタイミングで、被デバッグプログラムを動作させ、試作機30の記憶資源の内容を読み書きする。この読み書きの制御は、シリアル転送ポート31、シリアル回線40、専用端子39を介してモニタ制御層61がモニタリングプログラム36と所定の通信シーケンスを行うことにより行われる。これらの通信シーケンスの実行条件とは、様式変換ルーチン74による関数呼出が行われることである。

【0064】例えばメモリの内容を読み出すための通信シーケンスは、図15の(2)に示すパケットを、(1)のプロトコルに従って送受信することにより行われる。その通信シーケンスがなされるのは、モニタ制御層61が関数呼出コードMON_get_mem(adr,length,unit,image)を用いて関数呼び出しを行った場合である。図15の(2)に示した3種のパケット、即ちMEMREADパケット、ACKパケット、DATA1/n~n/nパケットは何れも8Byte長フィールドを有する。

【0065】MEMREADパケットの第0バイトはCMDフィールドであり、コマンドの種別を示すコマンドコードが書

き込まれる。第1バイトはUITフィールドであり、メモリからの読み出し単位(バイト、ワード、ダブルワードがある)をプロセッサに指示する。第2、第3バイトは、ADRSフィールドであり、読み出すべきメモリデータのサイズが書き込まれる。第4～第7バイトにメモリの読み出先アドレスが書き込まれる。

【0066】ACKパケットは、第0バイトにモニタリングプログラム36において発生したエラーの種別を示す種別コードが書き込まれる。図17は、メモリ内容の読み出しシーケンスを実行する際のモニタ制御層61の処理内容を示すフローチャートである。様式変換ルーチン74が関数呼出コードMON_get_mem(adr,length,unit,image)にて関数呼び出しどと、モニタ制御層61は、ステップS1においてCMDフィールド、sizeフィールド、ADRSフィールド、UITフィールドを設定し、送信パケットを作成する。この際、どのような内容を各フィールドに書き込むかであるが、CMDフィールドに対しては、MEMREADを示すコードを書き込む。sizeフィールドに対しては、引数lengthに示されている読み出しサイズを書き込み、ADRSフィールドに対しては、引数adrに示されている読み出し先アドレスを書き込む。UITフィールドに対しては、引数unitに示されている読み出し先アドレスを書き込む。このようにしてMEMREADパケットを作成すると、ステップS2においてシリアル回線40を介して、作成したパケットをモニタリングプログラム36側に送信する。送信後、モニタ制御層61は、ステップS3においてACKパケットの受信待ちを行う。ACKパケットが送信されれば、ステップS3がYesとなり、ステップS4に移行する。ステップS4では、モニタ制御層61はACKパケットのERRフィールドをチェックする。もし異常ならば、ステップS2に移行してMEMREADパケットの再発行を行うが、正常ならば、ステップS5において変数iを1に設定し、ステップS6において引数に指定されたlengthをDATAパケットサイズ(8byte)で割った数に変数nを設定する。変数iとは、各DATAパケットに個別に付された番号を示し、変数nとは、DATAパケットの総数を示す。

【0067】これらの変数に初期値を設定すると、ステップS7においてモニタ制御層61は受信パケットi(i=1,2,3,4~n)の受信待ちを行う。このように受信待ちを行っている間にDATAパケットが送信されれば、モニタ制御層61はこれを受信する。そしてステップS8において受信したDATAパケットのiのDATAフィールドに記載されている8バイトデータを共用ワークエリア77内の引数imageで指定されている領域に格納する。格納後、ステップS9において変数iと変数nとを比較し、もし変数iと変数nと異なるなら、ステップS10において変数iをインクリメントしてステップS7に移行する。

【0068】以降、ステップS9がYes、即ち、変数i=変数nとなるまでステップS7におけるDATAパケットの

受信と、ステップS 8における共用ワークエリア77に對しての8バイトデータの格納とを繰り返す。ステップS 7～ステップS 10の繰り返しにより、試作機30のデータはシリアル転送ポート31～シリアル回線40～専用端子39を介して8バイトずつモニタ制御層61に転送されてゆく。転送された8バイトのデータは共用ワークエリア77における領域imageに蓄積される。このような繰り返しがn回行われると、関数呼出コードに引数lengthで指定したサイズのデータが共用ワークエリア77のimage領域上に格納される。

【0069】メモリ34上のデータが共用ワークエリア77上へと読み出されると、これをシュミレータ10及びICE制御層62に転送することが可能となる。上記のようにしてメモリ34上のデータが共用ワークエリア77上の領域imageへと読み出された後、様式変換ルーチン74が関数呼出コードS1M_set_mem(adr,length,unit,image)を用いて関数呼出を行うと、共用ワークエリア77において引数imageに格納されているメモリ34のターゲット内ワークエリア38のデータがメモリ模擬ルーチン96内の開始アドレスadr以降の領域に書き込まれる。

【0070】同様の通信シーケンスを、残りのレジスタの内容、ブレークポイント、割込信号発生要件情報、周辺I/Oレジスタの内容、トレースデータについて繰り返し、モニタ制御層61は試作機30の記憶資源の状態情報をメモリ模擬ルーチン96へと転送する。またターゲット内ワークエリア38の内容を試作機30内のメモリ34に書き込むための通信シーケンスは、図16の(2)に示すパケットを、(1)のプロトコルに従って送受信することにより行われる。通信シーケンスがなされるのは、モニタ制御層61が関数呼出コードMON_set_mem(adr,length,unit,image)を用いて関数呼び出しを行った場合である。

【0071】図16の(2)に示した3種のパケット、即ちMEMWRITEパケット、DATA1～nパケット、ACK1,2パケットは何れも8Byte長フィールドを有する。MEMWRITEパケットの第0バイトは、CMDフィールドであり、コマンドの種別を示すコマンドコードが書き込まれる。第1バイトはUITフィールドであり、メモリからの読み出し単位(バイト、ワード、ダブルワードがある)をプロセッサに指示する。第2、第3バイトはSIZEフィールドであり、読み出すべきターゲット内ワークエリア38データのサイズが書き込まれる。第4～第7バイトはADRSフィールドであり、メモリ34の内容の書込先アドレスが書き込まれる。

【0072】ACK1パケット、ACK2パケットは、第0バイトにモニタリングプログラム36において発生したエラーの種別を示す種別コードが書き込まれる。図18は、ターゲット内ワークエリア38の内容の読み出しシーケンスを実行する際のモニタ制御層61の処理内容を示す

フローチャートである。

【0073】様式変換ルーチン74が関数呼出コードMON_set_mem(adr,length,unit,image)にて関数呼び出しそると、モニタ制御層61は、ステップS 11においてCMDフィールド、sizeフィールド、ADRSフィールドを設定し、送信パケットを作成する。この際、どのような内容を各フィールドに書き込むかであるが、CMDフィールドに対しては、MEMWRITEを示すコードを書き込む。sizeフィールドに対しては、引数lengthに示されているサイズを書き込み、ADRSフィールドに対しては、引数adrに示されている書込先アドレスを書き込む。UITフィールドに対しては、引数adrに示されている書込先アドレスを書き込む。

【0074】このようにしてMEMWRITEパケットを作成すると、ステップS 12においてシリアル回線40を介して、作成したパケットをモニタリングプログラム36側に送信する。送信後、モニタ制御層61は、ステップS 13においてACKパケットの受信待ちを行う。ACKパケットが送信されれば、ステップS 13がYesとなり、ステップS 14に移行する。ステップS 14では、モニタ制御層61はACK1パケットのERRフィールドをチェックする。もし異常ならば、ステップS 12に移行してMEMREADパケットの再発行を行うが、正常ならば、ステップS 15において変数jを1に設定し、ステップS 16において引数に指定されたlengthをDATAパケットサイズ(8byte)で割った数に変数nを設定する。変数jとは、各DATAパケットに個別に付された番号を示し、変数nとは、DATAパケットの送信総数を示す。

【0075】これらの変数に初期値を設定すると、ステップS 18において引数imageで指定された領域imageに格納されているデータのうち、(8*(j-1))バイト目から(8*j-1)バイト目までの8byteを読み出す。ここでj=1であるから、第0バイト目から第7バイト目までのデータが読み出され、モニタ制御層61はこれを送信DATAパケットjのDATAフィールドに記載する。記載後、ステップS 19においてDATAパケットを発行してステップS 20に移行する。ステップS 20において変数jと変数nとを比較し、もし変数jと変数nと異なるなら、ステップS 21において変数jをインクリメントしてステップS 18に移行する。

【0076】以降、ステップS 20がYes、即ち、変数j=変数nとなるまでステップS 18、ステップS 19における8バイトデータの送信とを繰り返す。ステップS 18～ステップS 21の繰り返しにより、共用ワークエリア77のデータはシリアル転送ポート31、シリアル回線40、専用端子39を介して8バイトずつメモリ34に転送されてゆく。転送された8バイトのデータはメモリ34に蓄積される。このような繰り返しがn回行われると、領域imageに格納されているデータがメモリ34上に格納される。

【0077】同様の通信シーケンスを、残りのレジスタの内容、ブレークポイント、割込信号発生要件情報、周辺I/Oレジスタの内容、トレースデータについて繰り返し、モニタ制御層62は共用ワークエリア77の状態情報を試作機30の記憶資源に格納する。ここで関数呼出コードMON_set_mem(addr, length, unit, image)を用いて関数呼出を行う前に予め共用ワークエリア77にシュミレータ10の保持値を書き込んでおけば、共用ワークエリア77上のシュミレータ10及びICE制御層62の保持値をメモリ34に転送することも可能である。

【0078】ICE制御層62は、評価ボード20におけるICEチップ23の制御権を掌握している。即ち、被デバッグプログラムの管轄権及び評価ボード20上のメモリ、レジスタ等記憶資源のアクセス権を有する。そして、所望のタイミングで、被デバッグプログラムを動作させ、評価ボード20の記憶資源に共用ワークエリア77の内容を書き込ませる。

【0079】上記読み書き制御は、様式変換ルーチン74による、ICE型を明示した関数呼出に伴って、ICE制御層62がデバイスドライバ21にBIOSコマンドを発行することにより行われる。図19は、評価ボード20内の被デバッグプログラム37の内容を読み出させる際の、ICE制御層62の制御内容を示す説明図である。ICE制御層62が関数呼出コードICE_get_mem(addr, length, unit, image)を用いて関数呼び出しされると、ICE制御層62はデバイスドライバ21に対して、BIOSコマンド『MEM_READ(addr, type, num)』を発行する。

【0080】BIOSコマンドのパラメータについて説明する。パラメータaddrは、評価ボード20のワークエリアのうち、どのアドレスから状態情報を読み出すかを示し、パラメータtypeは、評価ボード20上での読み出し単位を示す。引数unitも同内容の読み出し単位を指定しているが、評価ボード20で通用している取得単位が異なる場合がある。その場合、デバイスドライバ21は、評価ボード20で通用している取得単位にパラメータtypeを設定する。

【0081】パラメータnumは、語数を示す。ここでいう『語』の語長は、パラメータtypeで指定された取得単位を意味し、パラメータnumは関数呼出時において、引数lengthにおいて指定した読み出しサイズを語長unitで割った数に設定される。以上のようにして、パラメータaddr, type, numを設定すると、デバイスドライバ21にMEM_READ(addr, type, num)を発行する。発行後、デバイスドライバ21が結果を出力するのを待ち、結果を出力すると、その内容を評価ボード20におけるメモリ24の領域のaddrからの値として受け取り、これを共用ワークエリア77の引数addrに指定された領域に格納する。

【0082】図20は、評価ボード20内の被デバッグプログラム37に内容を書き込む際の、ICE制御層62の制御内容を示す説明図である。ICE制御層62が関数

呼出コードICE_set_mem(addr, length, unit, image)を用いて関数呼び出しされると、ICE制御層62はデバイスドライバ21に対して、BIOSコマンド『MEM_WRITE(addr, type, num)』を発行する。発行後、共用ワークエリア77内の引数addrに指定されている領域の値を、書き込むべき値としてデバイスドライバ21に出力する。

【0083】当該BIOSコマンドのパラメータの内容について説明する。パラメータaddrは、共用ワークエリア77の内容を評価ボード20内のワークエリア38のうちどのアドレスに書き込むかを示し、パラメータtypeは、書き込み単位を示す。パラメータnumは、語数を示す。ここでいう『語』の語長は、評価ボード20内の取得単位を意味し、関数呼出時において、引数lengthにおいて指定した書き込みサイズを語長typeで割った数に設定される。

【0084】以上のようにして、パラメータaddr, type, numを設定すると、デバイスドライバ21にMEM_WRITE(addr, type, num)を発行する。デバイスドライバ21は、ICE制御層62が発行したBIOSコマンドに従って、評価ボード20上のアドレス設定用ICE資源、サイズ設定用ICE資源、動作指定用ICE資源の設定を行う。

【0085】これらのICE用資源を設定する理由は、ICEチップ23は、これらのICE資源に記述されている内容をホストコンピュータ1からの指示と解釈して、メモリの読み書きやブレークポイント設定等の動作を行うからである。BIOSコマンドMEM_READ(addr, type, num)が発行されればデバイスドライバ21は、第1にパラメータaddrに指定された値をアドレス設定用ICE資源に設定し、第2にパラメータnumに設定された値をサイズ設定用ICE資源に設定する。そして、動作指定用ICE資源にリード動作を設定する。そうするとICEチップ23は、メモリ24内の領域のうち、パラメータaddrに指定された内容をnumだけ、デバイスドライバ21へと読み出す。デバイスドライバ21は読み出された値を評価ボード20内のメモリ24の状態情報としてICE制御層62に出力する。

【0086】BIOSコマンドMEM_WRITE(addr, type, num)が発行されればデバイスドライバ21は、第1にパラメータaddrに指定された値をアドレス設定用ICE資源に設定し、第2にパラメータnumに設定された値をサイズ設定用ICE資源に設定する。そして、動作指定用ICE資源にライト動作を設定する。そしてデバイスドライバ21がモニタ制御層61から渡された共用ワークエリア77内の領域imageの値を出力するとICEチップ23は、メモリ24内の領域のうち、パラメータaddrに指定された領域にnumだけ、デバイスドライバ21から引き渡された内容を書き込む。

【0087】以上までの構成要素による本デバッグ装置の動作について説明する。図12は、ユーザインターフェイス層50により行われるメッセージ表示の一例を示す図である。デバッグ装置が起動すると、ユーザインタ

一フェイス層50は先ず、参照符号p1に示すように、『*ハードウェア環境は{TARGET1}に設定されています。{TARGET1}は{モニタ型}のハードウェア環境です。』というメッセージを表示し、『*コマンドを入力して下さい』というメッセージと、プロンプトとを表示してコマンドの入力待ちとなる。

【0088】<『readMEM 0x0100,16』の入力>この状態で操作者がキーボード3をタイプして『readMEM 0x0100,16』というコマンドを入力すると、ユーザインターフェイス層50は、入力されたコマンドをデバッグ核層51に出力して、当該コマンドの解釈及び実行を委ねる。コマンド変換ルーチン73は入力されたコマンドを関数呼出コード『TCI_read_MEM (0x0100,16, byte, image)』に変換して、様式変換ルーチン74に出力する。様式変換ルーチン74は、関数呼出コードをハードウェア環境に依存した様式に書き換える。ハードウェア環境はモニタ型であるから、モニタ型に依存した様式『MON_read_MEM (0x0100,16, byte, image)』に書き換えられる。

【0089】様式変換ルーチン74が関数呼出コードMON_get_mem(0x0100,16, byte, image)にて関数呼び出しそると、モニタ制御層61は、図17のフローチャートに従い、試作機30のデータをシリアル転送ポート31-シリアル回線40-専用端子39を介して8バイトずつモニタ制御層61に読み出す。読み出された8バイトのデータは共用ワークエリア77における領域imageに蓄積される。このような繰り返しがn回行われると、関数呼出コードに引数lengthで指定したサイズのデータが共用ワークエリア77のimage領域上に格納される。

【0090】ここで共用ワークエリア77の格納内容が、『FF FF FF FF FF FF FF FFFF FF FF FF FF FF FF』であると、ユーザインターフェイス層50は、『*ハードウェア環境 {TARGET1} の {0x0100,16} の値は以下の通りです』という文字列を表示して、操作者が指定した読み出先をエコーバックすると共に、共用ワークエリア77の保持値『FF FF FFFF FF FFFF FF』を表示する。その後、『*コマンドを入力して下さい』というメッセージを表示してコマンドの入力待ちとなる。

【0091】<『writeREG D1,03』の入力>操作者がキーボード3をタイプして『writeREG D1,03』というコマンドを入力すると、ユーザインターフェイス層50は、入力されたコマンドをデバッグ核層51に出力して、当該コマンドの解釈及び実行を委ねる。コマンド変換ルーチン73は入力されたコマンドに従って、共用ワークエリア内の配列regsのレジスタD1の要素に03を格納し、関数呼出コード『TCI_write_REG (regs, mask)』に変換して、様式変換ルーチン74に出力する。様式変換ルーチン74は、関数呼出コードをハードウェア環境に依存した様式に書き換える。ハードウェア環境はモニタ型であるから、モニタ型に依存した様式『MON_write_REG (D1,

03)』に書き換えられる。

【0092】様式変換ルーチン74が関数呼出コードMON_write_REG(regs, mask)にて関数呼び出しそると、モニタ制御層61は、レジスタ書き込みをプロセッサ33に指示すると共に即値『03』をシリアル転送ポート31-シリアル回線40-専用端子39を介して試作機30に出力する。レジスタ書き込みが命じられたので、プロセッサ33は、当該即値を指示されたレジスタD1に書き込む。

【0093】その後、正常書き込みを示す信号をプロセッサ33がモニタ制御層61に出力すると、モニタ制御層61は正常終了を、様式変換ルーチン74-コマンド変換ルーチン73-コマンドインタプリタ72を介してユーザインターフェイス層50に通知する。ユーザインターフェイス層50は『*ハードウェア環境 {TARGET1} の {D1} に値 {03} を設定しました』というメッセージによりエコーバックを行った後、プロムプト『>』を表示して、操作者からの入力待ちになる。

【0094】<『SetBP BP1,0x0123』の入力>操作者がキーボード3をタイプして『SetBP BP1,0x0123』というコマンドを入力すると、ユーザインターフェイス層50は、入力されたコマンドをデバッグ核層51に出力して、当該コマンドの解釈及び実行を委ねる。コマンド変換ルーチン73は入力されたコマンドを関数呼出コード『TCI_SET_BP (BP1,0x0123)』に変換して、様式変換ルーチン74に出力する。様式変換ルーチン74は、関数呼出コードをハードウェア環境に依存した様式に書き換える。ハードウェア環境はモニタ型であるから、モニタ型に依存した様式『MON_SET_BP (BP1,0x0123)』に書き換えられる。

【0095】様式変換ルーチン74が関数呼出コードMON_SET_BP (BP1,0x0123)にて関数呼び出しそると、モニタ制御層61は、アドレス『0x0123』にブレークポイント『BP1』を設定する旨をシリアル転送ポート31-シリアル回線40-専用端子39を介して試作機30に出力する。ブレークポイント設定が命じられたので、プロセッサ33は、当該アドレス0x0123に、ブレークポイントBP1を設定する。

【0096】その後、正常設定を示す信号をプロセッサ33がモニタ制御層61に出力すると、モニタ制御層61は正常終了を、様式変換ルーチン74-コマンド変換ルーチン73-コマンドインタプリタ72を介してユーザインターフェイス層50に通知する。正常設定が通知されると、ユーザインターフェイス層50は『*ハードウェア環境 {TARGET1} の {BP1} に値 {0x0123} を設定しました』というメッセージを表示した後、プロムプト『>』を表示して、操作者からの入力待ちになる。

【0097】<『RUN』の入力>操作者がキーボード3をタイプして『RUN』というコマンドを入力すると、ユーザインターフェイス層50は、入力されたコマンドを

デバッグ核層51に出力して、当該コマンドの解釈及び実行を委ねる。コマンド変換ルーチン73は入力されたコマンドを関数呼出コード『TCI_RUN(0x0000)』に変換して、様式変換ルーチン74に出力する。様式変換ルーチン74は、関数呼出コードをハードウェア環境に依存した様式に書き換える。ハードウェア環境はモニタ型であるから、モニタ型に依存した様式『MON_RUN(0x0000)』に書き換えられる。

【0098】様式変換ルーチン74が関数呼出コードMON_RUN(0x0000)にて関数呼び出しづると、モニタ制御層61は、被デバッグプログラム37を実行する旨をシリアル転送ポート31シリアル回線40専用端子39を介して試作機30に出力する。プログラム実行が命じられたので、プロセッサ33は、被デバッグプログラム37の実行を開始する。

【0099】プロセッサ33が被デバッグプログラム37の実行を開始したので、試作機30は動作を開始する。その後、先に設定されたブレークポイントにより、被デバッグプログラム37の動作が中断すると、プロセッサ33はプログラムが中断した旨を専用端子39シリアル回線40シリアル転送ポート31を介してモニタ制御層61に通知する。

【0100】動作中断を示す信号をプロセッサ33がモニタ制御層61に出力すると、モニタ制御層61は動作中断を、様式変換ルーチン74コマンド変換ルーチン73コマンドインタプリタ72を介してユーザインターフェイス層50に通知する。動作中断が通知されると、ユーザインターフェイス層50は『* {TARGET1} のプログラムは {0x0123} で中断しました。』というメッセージを表示した後、プロムプト『>』を表示して、操作者からの入力待ちになる。

【0101】<selectTARGETの入力時の動作>図21～図26は、図7及び図8に示したデバッグ装置の内部階層図に対して、コマンド、関数呼出コード、状態情報の行き来を加筆した説明図である。そのうち図21は、モニタ型ハードウェア環境におけるプログラムの動作中に切り換え先をシミュレータ型に規定したSelectTargetコマンドが入力された場合の状態情報の行き来を示している。各階層にはこの際に用いられる関数呼出コード、コマンドが配置され、これらの間の行き来が矢印にて加筆されている。図21の説明図と、図30のフローチャートを参照しながらselectTARGETコマンドの解説に関連するコマンドインタプリタ72の処理内容を説明する。

【0102】モニタ型のハードウェア環境にて動作検証を行っている際、操作者がキーボード3をキータイプして、切換先をシミュレータに規定したselectTARGETコマンドを入力したものとする。図30におけるステップS81においてユーザインターフェイス層50は図12に示したプロムプト『>』を表示し、操作者からのコマンド入力を待っている。キータイプにより操作者がデバッ

グコマンドを入力すると、ステップS82に移行して、コマンドインタプリタ72は入力されたコマンドの解析を行う。解析の結果オペコードが『selectTARGET』ならステップS83がYesとなってステップS84に移行し、現在デバッグに用いているハードウェア環境（以下『src_target』という）側にプログラム実行の停止を指示する。そしてステップS85、ステップS86、ステップS87、ステップS88、ステップS89、ステップS90においてコマンドインタプリタ72はcopyMEMコマンド、copyREGコマンド、copyBPコマンド、copyINTコマンド、copyI/Oコマンド、copyTRACE_DATAコマンドを自動発行する（図21の矢印y1参照）。

【0103】copyMEM～copyTRACE_DATAコマンドの自動発行が行われると、下位層に位置するコマンド変換ルーチン73は、自動発行されたコマンドを関数呼出コードに変換する。サブコマンドCopyMEMは、コマンド変換ルーチン73によりTCI_get_mem(adr, length, unit, image)という関数呼出コードと、TCI_set_mem(adr, length, unit, image)という関数呼出コードとに変換される（図21の矢印y2参照）。サブコマンドCopyRegは、コマンド変換ルーチン73によりTCI_get_reg(regs, mask)といい関数呼出コードと、TCI_set_reg(regs, mask)といい関数呼出コードとに変換される。

【0104】様式変換ルーチン74は、コマンド変換ルーチン73により変換された関数呼出コードをハードウェア環境に依存した様式に変換する。切換元はモニタ型に設定され、切換先はシミュレータ型に設定されていることから、先ず様式変換ルーチン74は、TCI_get_mem(adr, length, unit, image)における『TCI_get_mem』という関数名を切換元を明示した関数名に書き換える。切換元のハードウェア環境はモニタ型であるから、モニタ型を明示した関数名『MON_get_mem』に書き換え、MON_get_mem(adr, length, unit, image)を得る（図21の矢印y3参照）。

【0105】続いて様式変換ルーチン74は、TCI_set_mem(adr, length, unit, image)における『TCI_set_mem』という関数名を切換先を明示した関数名に書き換える。切換先のハードウェア環境はシミュレータ型であるから、シミュレータ型を明示した関数名『SIM_set_mem』に書き換える。書き換え結果は、SIM_set_mem(adr, length, unit, image)となる（図21の矢印y4参照）。関数名を書き換えた後、様式変換ルーチン74は関数呼出コードを用いて関数呼出を行う。

【0106】様式変換ルーチン74が関数呼出コードMON_get_mem(adr, length, unit, image)にて関数呼び出しづると、モニタ制御層61は、図17のフローチャートに従い、試作機30のデータをシリアル転送ポート31シリアル回線40専用端子39を介して8バイトずつモニタ制御層61に読み出す。読み出された8バイトのデータは共用ワークエリア77における領域imageに蓄

積される。このような繰り返しがn回行われると、関数呼出コードに引数lengthで指定したサイズのデータが状態情報として共用ワークエリア77のimage領域上に格納される（矢印T1参照）。

【0107】蓄積された後、様式変換ルーチン74が関数呼出コードSIM_set_mem(adr,length,unit,image)を用いて関数呼出を行う。様式変換ルーチン74により呼び出されれば、共用ワークエリア77において引数imageに指定されている領域のデータを状態情報としてメモリ模擬ルーチン96内の開始アドレスadr以降の領域に書き込む（矢印T2参照）。書き込み作業は取得長lengthに指定されているサイズだけ、取得単位unitにて指定されている単位で行われる。このように共用ワークエリア77からメモリ模擬ルーチン96への書き込みを行うことにより試作機30からシミュレータ10へのメモリ内容の転送が完了する。

【0108】同様の作業をcopyREGコマンド、copyBPコマンド、copyINTコマンド、copyI/Oコマンド、copyTRAC_E_DATAコマンドについて行う。そうすると、試作機30の記憶資源の動作状態がシミュレータ10へと転送される。その結果、それまでの試作機30の動作状態がシミュレータ10に継承される。継承後、動作状態情報が設定されると、プログラムの動作を切換先シミュレータ10において再開するよう制御する。

【0109】図22は、モニタ型ハードウェア環境におけるプログラムの動作中に切り換え先をICE型に規定したSelectTargetコマンドが入力された場合の状態情報の行き来を示している。本図が図21と異なるのは、ICE型ハードウェア環境を指定したselectTARGETコマンドが入力されたため、コマンド変換ルーチン73が生成した関数呼出コードTCI_SET_mem(adr,length,unit,image)を様式変換ルーチン74がICE型に依存した様式に書き換え、この書き換えにより、ICE制御層62が駆動された点である。

【0110】駆動されたICE制御層62は、試作機30内の状態情報が格納された領域imageの内容をBIOSコマンドを用いて評価ボード20内に書き込む（矢印T4参照）。このようにして、試作機30の状態情報が評価ボード20へと転送され、それまでの試作機30のプログラムの動作が評価ボード20上へと継承される。図23は、ICE型ハードウェア環境におけるプログラムの動作中に切り換え先をモニタ型に規定したSelectTargetコマンドが入力された場合の状態情報の行き来を示している。本図が図22と異なるのは、転送元と転送先の関係が入れ代わっている点である。即ち、転送元がICE型であり、転送先がモニタ型なのである。ICE型ハードウェア環境を指定したselectTARGETコマンドが入力されたため、コマンド変換ルーチン73が生成した関数呼出コードTCI_GET_mem(adr,length,unit,image)を様式変換ルーチン74がICE型に依存した様式に書き換える。この

書き換えにより、ICE制御層62が駆動され、駆動されたICE制御層62は、評価ボード20内の状態情報をBIOSコマンドを用いて共用ワークエリア77内の領域image内に書き込む（矢印T5参照）。

【0111】一方コマンド変換ルーチン73が生成した関数呼出コードTCI_SET_mem(adr,length,unit,image)を様式変換ルーチン74がモニタ型に依存した様式に書き換える。この書き換えにより、モニタ制御層61が駆動され、駆動されたモニタ制御層61は、評価ボード20内の状態情報が格納された領域imageの内容をシリアル回線40を介して試作機30内に書き込む（矢印T6参照）。このようにして、評価ボード20の状態情報が試作機30へと転送され、それまでの評価ボード20のプログラムの動作が試作機30上へと継承される。

【0112】図24は、ICE型ハードウェア環境におけるプログラムの動作中に切り換え先をシミュレータ型に規定したSelectTargetコマンドが入力された場合の状態情報の行き来を示している。本図が図23と異なるのは、転送先がモニタ型からシミュレータ型に入れ代わっている点である。即ち、転送元がICE型であり、転送先がシミュレータ型なのである。ICE型ハードウェア環境を指定したselectTARGETコマンドが入力されたため、コマンド変換ルーチン73が生成した関数呼出コードTCI_GET_mem(adr,length,unit,image)を様式変換ルーチン74がICE型に依存した様式に書き換える。この書き換えにより、ICE制御層62が駆動され、駆動されたICE制御層62は、評価ボード20内の状態情報をBIOSコマンドを用いて共用ワークエリア77内の領域image内に書き込む（矢印T7参照）。

【0113】一方コマンド変換ルーチン73が生成した関数呼出コードTCI_SET_mem(adr,length,unit,image)を様式変換ルーチン74がシミュレータ型に依存した様式に書き換える。この書き換えにより、シミュレータ10が駆動され、評価ボード20内の状態情報が格納された領域imageの内容をその内部のメモリ模擬ルーチン96に書き込む（矢印T8参照）。このようにして、評価ボード20の状態情報がシミュレータ10へと転送され、それまでの評価ボード20のプログラムの動作がシミュレータ10上へと継承される。

【0114】図25は、シミュレータ型ハードウェア環境におけるプログラムの動作中に切り換え先をモニタ型に規定したSelectTargetコマンドが入力された場合の状態情報の行き来を示している。本図が図21と異なるのは、転送先と転送元とが入れ代わっている点である。即ち、転送元がシミュレータ型であり、転送先がモニタ型なのである。モニタ型ハードウェア環境を指定したselectTARGETコマンドが入力されたため、コマンド変換ルーチン73が生成した関数呼出コードTCI_GET_mem(adr,length,unit,image)を様式変換ルーチン74がシミュレータ型に依存した様式に書き換える。この書き換えによ

り、シュミレータ10が駆動され、駆動されたシュミレータ10は、自身の状態情報を共用ワークエリア77内の領域image内に書き込む（矢印T9参照）。

【0115】一方コマンド変換ルーチン73が生成した関数呼出コードTCI_SET_mem(adr,length,unit,image)を様式変換ルーチン74がモニタ型に依存した様式に書き換える。この書き換えにより、モニタ制御層61が駆動され、駆動されたモニタ制御層61は、格納された領域imageの内容を試作機30内部に書き込む（矢印T10参照）。このようにして、シュミレータ10の状態情報が試作機30へと転送され、それまでのシュミレータ10のプログラムの動作が試作機30上へと継承される。

【0116】図26は、シュミレータ型ハードウェア環境におけるプログラムの動作中に切り換え先をICE型に規定したSelectTargetコマンドが入力された場合の状態情報の行き来を示している。本図が図22と異なるのは、転送先と転送元とが入れ代わっている点である。即ち、転送元がシュミレータ型であり、転送先がICE型なのである。ICE型ハードウェア環境を指定したselectTAR GETコマンドが入力されたため、コマンド変換ルーチン73が生成した関数呼出コードTCI_GET_mem(adr,length,unit,image)を様式変換ルーチン74がシュミレータ型に依存した様式に書き換える。この書き換えにより、シュミレータ10が駆動され、駆動されたシュミレータ10は、自身の状態情報を共用ワークエリア77内の領域image内に書き込む（矢印T11参照）。

【0117】一方コマンド変換ルーチン73が生成した関数呼出コードTCI_SET_mem(adr,length,unit,image)を様式変換ルーチン74がICE型に依存した様式に書き換える。この書き換えにより、ICE制御層62が駆動され、駆動されたICE制御層62は、格納された領域imageの内容を評価ボード20内部に書き込む（矢印T12参照）。このようにして、シュミレータ10の状態情報が評価ボード20へと転送され、それまでのシュミレータ10のプログラムの動作が評価ボード20上へと継承される。

【0118】モニタ型、ICE型、シュミレータ型間のハードウェア環境の任意の切り換えは、以上の図21～図26に示した状態情報の受け渡しにより実現される。統いて、説明を保留したターゲット相互接続層52の残りの構成要素（ワークエリアコンテンツ解析モジュール75）について説明する。ワークエリアコンテンツ解析モジュール75は、各ハードウェア環境が有するメモリの領域を、ブレークポイントの設定が可能な領域と不可能な領域とに分割してハードウェア環境毎に管理するブレークポイント管理機能(1)と、各ハードウェア環境がトレースデータの蓄積に用いることができるメモリサイズをハードウェア環境毎に管理するトレースデータ蓄積サイズ管理機能(2)と、各ハードウェア環境が割込発生条件の成立/不成立を監視する能力を有するか否かをハ

ドウェア環境毎に管理する割込発生条件管理機能(3)と、I/Oマッピング情報の管理機能(4)とを有する。

【0119】ワークエリアコンテンツ解析モジュール75が何故ブレークポイント管理機能(1)を有するかであるが、これはブレークポイントを割込命令の記入により行う場合、ブレークポイント設定の可否がICE型、モニタ型、シュミレータ型間で異なることを想定しているからである。即ち切換元でRAMで構成されブレークポイントが設定されていたメモリ空間内の所定領域が、切換先でROMで構成されている場合、そのブレークポイントを切換先において設定することは不可能である。このように切換元にて設定可能なブレークポイントが切換先では不可能となる場合、これを予め操作者に通知しておかないと、ハードウェア環境を切り換えた途端にブレークポイントが無効になるという現象が発生する。このようにハードウェア環境の切り換えに伴いブレークポイントが無効になると、操作者はこれを故障と勘違いする恐れが有り、不信感を抱かせかねない。そこでワークエリアコンテンツ解析モジュール75にメモリ内のどの領域にブレークポイント設定が可能であり、どの領域はブレークポイント設定が不可能であるかを管理させておくのである。図28(a)はブレークポイント管理に用いる管理情報の一例を示す図である。図中の『START_ADDR1～END_ADDR1』、『START_ADDR2～END_ADDR2』、『START_ADDR3～END_ADDR3』は、それぞれがブレークポイント設定の設定が可能な領域の開始アドレス、終了アドレスを示している。図中の『VALID_AREA1』、『VALID_AREA2』、『VALID_AREA3』は、ブレークポイントの設定可能領域に付されたラベルを示している。

【0120】次にワークエリアコンテンツ解析モジュール75が何故トレースデータ蓄積サイズ管理機能(2)を有するかであるが、これはトレースデータの蓄積に割り当つてが可能であるメモリサイズがモニタ型、ICE型、シュミレータ型間で異なることを想定しているからである。即ち切換元がシミュレータであり、切換先がモニタである場合、切換元側で蓄積された全てのトレースデータをそのまま切換先に転送するのは不可能である。転送途中で切換先のメモリがオーバーフローし、これまで蓄積してきたトレースデータの一部が欠落してしまう。このような欠落の可能性は予め操作者に通知しておかないと、操作者はトレースデータが何の前触れもなく欠落するのを見て不信感を抱いてしまう。そこでワークエリアコンテンツ解析モジュール75はトレースデータの蓄積に割り当つてが可能であるメモリサイズをハードウェア環境毎に管理するのである。図28(b)は、トレースデータを蓄積するためのメモリサイズを管理するための管理情報の一例を示す図である。図中の『target1』、『target2』は、各ハードウェア環境の識別名であり、『SPACE_FOR_TRACE_DATA_MAX 16Kbyte』は、各々のハードウェア環境内においてトレースデータの蓄積に割り当てる

ことができるメモリサイズが16Kbyteであることを示している。

【0121】次にワークエリアコンテンツ解析モジュール75が何故割り込み発生条件管理機能(3)を有するかであるが、これは割り込み発生条件が成立しているか、不成立であるかの監視能力の有無がモニタ型、ICE型、シュミレータ型間で異なることを想定しているからである。例えば、割り込み発生条件が命令の実行時間であり、切換元及び切換先がシミュレータ、モニタであるものとする。この場合切換元側では命令実行のカウントをソフトウェアにより実現すれば良いから割り込み発生条件の成立を監視することができる。これに対して切換先側では命令実行のカウントは、特別のハードウェアの改造を行わない限り実現することは不可能である。もしそのような改造がなされていない場合、切換先では割り込み発生条件の成立を監視することは不可能となるので、切換先では何時まで立っても割り込み発生条件が発生しないという現象が表れる。このように割り込み発生条件が成立しない場合は前もって操作者に通知しておかないと、操作者に不信感を抱せてしまう。そこでワークエリアコンテンツ解析モジュール75は割り込み発生条件の監視能力の有無をハードウェア環境毎に管理するのである。図28(c)は、ハードウェア環境毎の監視能力を管理するための管理情報の一例を示す図である。図中の『target1』『target2』は、各ハードウェア環境の識別名であり、『CYCLE_NUMBER_COUNT VALID』は、対応する識別情報のハードウェア環境において、プロセッサのサイクル数のカウントが可能である旨を示し、『CYCLE_NUMBER_COUNT INVALID』は、対応する識別情報のハードウェア環境において、プロセッサのサイクル数のカウントが不可能である旨を示す。

【0122】次にワークエリアコンテンツ解析モジュール75が何故I/Oマッピング情報の管理機能を有するかであるが、これは周辺I/Oの実装状況がモニタ型、ICE型、シュミレータ型間で異なることを想定しているからである。即ち切換先がシミュレータであり、切換元がモニタであるものとする。この場合、切換元では、ハードウェア調整等の進捗が遅れている等の理由により、周辺I/Oが一部しか動作していないが、切換先側では実行環境をソフトウェアにより仮想的に作成しているため周辺I/Oを動作させることができる。この状態のままI/Oの状態情報を転送すると、切換先の周辺I/Oには、一部無効な値が格納されてしまう。そこでワークエリアコンテンツ解析モジュール75はハードウェア環境毎にどの周辺I/Oが動作可能であり、どの周辺I/Oが動作不可能であるかを個別に管理するのである。

【0123】ワークエリアコンテンツ解析モジュール75により管理されるI/Oマッピング情報の一例を図29に示す。I/Oマッピング情報とは、切換元ハードウェア環境において周辺I/Oレジスタの値が存在しない場合

に、切換先のハードウェア環境にどのような値を設定すれば良いかを示す情報である。図中の『I/O_REGISTER_A DDR1』は、一個目の周辺I/Oレジスタを示し、『PSW=STATUS1 0x1F』は、切換先のハードウェア環境においてプロセッサのPSWレジスタ値が『STATUS1』である場合に一個目の周辺I/Oレジスタに『0x1F』を設定する旨を示している。また『PSW=STATUS2 0x2A』は、切換先のハードウェア環境においてプロセッサのPSWレジスタの値が『STATUS2』である場合に一個目の周辺I/Oレジスタに『0x2A』を設定する旨を示している。更に『PSW=STATUS3 0x3B』は、切換先のハードウェア環境においてプロセッサのPSWレジスタの値が『STATUS3』である場合に一個目の周辺I/Oレジスタに『0x3B』を設定する旨を示している。

【0124】続いてselectTARGETコマンドの拡張型コマンドについて説明する。図10は拡張型コマンドのうち3つの代表的なものを示している。ここでいうselectTARGETコマンドの拡張とは、selectTARGETコマンドの発行に所定の条件を与えたという意味であり、図10における3つのコマンドはselectTARGETコマンド発行の条件がそれ異なる。

【0125】IFBreakコマンドは、現在デバッグに用いられているハードウェア環境に被デバッグプログラムの実行を指示し、もし当該実行がBreakNOで指定されたブレークポイントにより停止すれば、ターゲット相互接続層52にハードウェア環境の切り換えを行うようSelect TARGETを自動発行する。TargetASSIGNコマンドは、オペランドでTABLE*にて指定した割り当てにてハードウェア環境を切り換えるようターゲット相互接続層52に命じるコマンドである。図11は、TargetASSIGNコマンドにより指定されるテーブルの一例であり、ハードウェア環境切換の割り当てをターゲット相互接続層52に指示するようフォーマットが規定されている。本フォーマットでは、先頭アドレス、終了アドレスの一組毎に切り換えを望むハードウェア環境を指定できるようになっている。図11では先頭アドレス、終了アドレスの組みを最大n個の羅列することができ、それぞれの組み毎に異なるハードウェア環境を切り換え先に規定できる。

【0126】Switch()コマンドは、ハードウェア環境target0の変数varの値に応じてハードウェア環境を切り換えることをターゲット相互接続層52に命じるコマンドである。case'*1'は、変数varの値が*1である場合を想定しており、このときはハードウェア環境target1が起動される。case'*2'は、変数varの値が*2である場合を想定しており、このときはハードウェア環境target2が起動される。

【0127】続いて、図10に示したselectTARGETコマンドの3つの拡張コマンドが入力された場合のコマンドインタプリタ72の処理を図31～図33のフローチャートを参照しながら説明する。ユーザインターフェイス

層50に対して操作者が『TargetASSIGN』コマンドを入力したとする。この場合図30のステップS83がNoとなり、図31のステップS71に移行する。ステップS71においてコマンドインタプリタ72はオペコードがTargetASSIGNであるかを判定する。この場合コマンドのオペコードが合致するので、ステップS72においてTABLE*で指定されたテーブルを常駐させる。常駐後、ステップS73においてテーブルにおける各開始アドレス[i](i=1,2~n)にブレークポイントを設定する。設定後、ステップS74において被デバッグプログラムを実行させる。実行させると、ステップS75において被デバッグプログラムがブレークしたかを監視し、ブレークした場合はステップS76においてブレークしたアドレスが開始アドレス[i](i=1,2,3~n)のうち、何れに対応するかを判定する。何れかに該当すると判定すると、ステップS77において開始アドレス[i]に対応するハードウェア環境に切り換える旨のSelectTARGETを自動発行する。

【0128】次にユーザインターフェイス層50に対して操作者が『IFBreak』コマンドを入力したとする。この場合図30のステップS83がNoとなり、ステップS71に移行するが、図31のステップS71においてもNoとなり、ステップS91に移行する。ステップS91においてコマンドインタプリタ72は、入力されたコマンドのオペコードがIFBreakであるかを判定する。この場合Yesとなり、ステップS92に移行して、オペランドで指定された『BreakNO』『SelectTARGET target』を常駐させる。その後、ステップS93において被デバッグプログラムを実行させる。実行させると、ステップS94において被デバッグプログラムがブレークしたかを判定し、もしブレークしたなら、ステップS96においてブレークしたアドレスがブレークポイント[i](i=1,2,3~n)のうち、何れのBreakNOに対応するかを判定する。対応するものを判定すると、ステップS97においてSelectTARGETを自動発行する。

【0129】ユーザインターフェイス層50に対して操作者が『Switch(var)』コマンドを入力したとする。この場合図30のステップS83がNoとなり、ステップS71に移行するが、図31のステップS71においてもNoとなり、ステップS91に移行する。ステップS91においてもNoとなりステップS100へと進行する。ステップS100ではオペコードがSwitch(var)であるかが判定され、この場合はそうであるのでステップS101に移行する。ステップS101では、コマンドインタプリタ72は変数『var』をメモリアドレスmem_addr(var)に変換するようデバッグ核層51に指示し、変換後ステップS102において指定された『target0』のハードウェア環境に変数adr(var)のメモリデータの読み出しを指示する。指示後、ステップS103において読み出しされたメモリデータがcasei(i=1,2,3,4,5~n)の何れ

かの内容と一致するかを判定し、一致する場合は、ステップS104においてTARGET[i]に切り換える旨のSelectTARGETを自動発行する。

【0130】以上のように本実施形態によれば、select TARGETコマンド入力という手軽な操作により、操作者はデバッグ中のどのような状況においても、思い通りにハードウェア環境を切り換えることができる。そのためバグが発生した状況を他のハードウェア環境で再現する手間に煩わされることなく、別のハードウェア環境にてそのバグの発生原因を追究することができ、貴重な開発時間を有効に活用することができる。

【0131】特にバグが発生したデバッグ装置が解析機能に乏しいモニタである場合、そのバグ原因を追究をICEやシミュレータ等解析機能が豊富な実行環境に切り換えることができ、原因解明までの時間がより短縮される。上記実施形態に基づいて説明してきたが、現状において最善の効果が期待できるシステム例として提示したに過ぎない。本発明は、その要旨を逸脱しない範囲で変更実施することができる。ハードウェア環境の切り換えを意図したであればどのようなシステムに適用できることはいうまでもない。以下(a)(b)(c)に示すような変更実施が可能である。

【0132】(a) 本実施形態では、スーパーバイザプログラムをメモリに配したが、デバッグ核層51内に存在させる構成にしても良い。またターゲット相互接続層52は、デバッグ核層51内に存在する構成にしても良い。

(b) 以上の説明では、ICE型ハードウェア環境がブレークポイント、トレース設定、割り込み設定を有している場合について記述したが、切換先がブレークポイント、トレース設定、割り込み設定のうち何れかの機能が利用できない場合、利用できる機能の設定情報のみを切換先に設定するようにしてもよい。

【0133】(c) 以上の説明では、ターゲット内ワークエリア38の内容を全て切換先へと転送しているが、前回の転送分からの変更分のみを転送することにより転送負荷を軽減しても良い。この場合スタック領域と、外部変数領域の内容のバックアップをデバッグ核層51或はターゲット相互接続層52に記憶させておく。そしてもしselectTARGETコマンド入力によりハードウェア環境切り換えが指示されれば、そのバックアップしておいた記憶内容と、これから転送すべき記憶内容とを比較して変更された部分を検出し、変更があった部分のみを転送するのである。この場合、変更されたメモリ領域を記憶しておけば良いが、変更されたメモリ領域を含む領域を保存するようにしてもよい。

【0134】(d)(c)において変更されたメモリ領域をハードウェア環境の切替時に一括して転送する応用例を示したが、ハードウェア環境がメモリ内容を変更する毎に、変更された内容を別のハードウェア環境に転送

してもよい。この場合、一方のハードウェア環境を他のハードウェア環境に切替る際にメモリ内容の転送が発生しないので、ハードウェア環境の切り換えがより高速になる。

【0135】(e) さらに、以上の説明では、変更されたメモリ領域を記憶し、記憶されたメモリ領域を切換先に転送しているが、さらに変更されたレジスタの値を記憶するようにし、記憶されたレジスタの値を切換元から切換先に転送するようにしてもよい。この場合、変更されたレジスタの値のみを転送するので、全てのレジスタを転送する場合と比較してハードウェア環境切り換えがより高速となる。

【0136】(f) 被デバッグプログラムのブレーク時にハードウェア環境を切り換えるコマンドを示したが、ハードウェア環境がエリアブレークを使用できる時には、エリアブレークによるハードウェア環境の切替を実現してもよい。ここで、エリアブレークとは、あるアドレス領域にハードウェア環境がアクセスした時、実行中のプログラムを停止させる機能をさす。この場合、現在のプログラムの命令読み出し先がどの先頭アドレスと最終アドレスの範囲内にあるかを調べ、その範囲内に対応したプログラムを実行する。

【0137】(g) 本実施形態では、変数の値に応じてハードウェア環境を切り換えるというコマンドを示したが、パラメータの値に応じてハードウェア環境を切り換えるというコマンドを設けてハードウェア環境の切り換えを実現してもよい。

(h) ハードウェア環境の切換指示と、割込信号発生を監視する旨の指示と、切換先に規定すべきハードウェア環境の指示とをコマンド内に配置し、これらの指示をコマンドインタプリタ72が解読して、解読結果に基づいてハードウェア環境の切り換えを行わせてもよい。本デバッグ装置がこのような構成のコマンドよりハードウェア環境の切り換えを行うことにより、操作者は特定の割込信号発生時の被デバッグプログラムの動作を他のハードウェア環境において観測することができる。

【0138】(i) ハードウェア環境の切換指示と、ハードウェア環境内の所定メモリ或はレジスタの内容の書き換えを監視する旨の指示と、切換先に指定すべき第2のハードウェア環境の指示とをコマンド内に配し、これらの指示をコマンドインタプリタ72が解読して、解読結果に基づいてハードウェア環境の切り換えを行わせてもよい。本デバッグ装置がこのような構成のコマンドによりハードウェア環境の切り換えを行うことにより、操作者は特定メモリ或はレジスタの書き換え時の被デバッグプログラムの動作を他のハードウェア環境において観測することができる。

【0139】(j) ハードウェア環境の切換指示と、操作者が複数のハードウェア環境にどの実行予定時刻を対応付けたかを示す対応表とをコマンド内に配し、これら

の指示をコマンドインタプリタ72が解読して、解読結果に基づいてハードウェア環境の切り換えを行わせてもよい。

(k) デバッグ情報を、メモリ24及びメモリ模擬ルーチン96上に配置する構成を示したが、これらをデバッグ核層51～ターゲット相互接続層52上に配置してもよい。このように配置した場合、ターゲット依存層53内のモニタ制御層61、ICE制御層62、シミュレータからデバッグ情報を読み出す処理は実質不用となり、コマンド変換ルーチン73による関数呼出コードTCI_get_mem、TCI_get_reg、TCI_get_BPの発行は行わなくてもよい。

【0140】

【発明の効果】以上説明したようにデバッグ装置は、所定のターゲットマシン向けに開発された組み込みプログラムを当該ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち、何れかにおいてテスト動作させ、これの動作検証を行うデバッグ装置であって、前記ターゲットマシン、エミュレータマシン、ソフトウェアシミュレータは、組み込みプログラムの動作状態を示す動作状態情報を有し、動作状態情報の入出力を固有の様式にて行い、ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち何れかを動作検証対象として記憶する動作検証対象記憶手段と、操作者によるコマンド入力を受け付ける受付手段と、受付手段が受け付けたコマンド内に所定指示が含まれていればこれを検出する検出手段と、所定指示が検出されると、動作検証対象記憶手段に記憶されている動作検証対象を切換元動作検証対象に指定し、受付手段が受け付けたコマンドに基づいて前記ターゲットマシン、当該ターゲットマシンのエミュレータマシン、当該ターゲットマシンのソフトウェアシミュレータのうち切換元動作検証対象と異なるものを切換先動作検証対象に指定する指定手段と、切換元及び切換先が指定されると、切換元動作検証対象から動作状態情報を読み出す読出手段と、読み出された動作状態情報を、切換先動作検証対象の様式に変換する変換手段と、動作状態情報が切換先の様式に変換されれば、様式が変換された動作状態情報を切換先動作検証対象に設定する設定手段と、動作状態情報が設定されると、組み込みプログラムの動作を切換先動作検証対象において再開するよう制御する動作再開手段とを備えるので、プログラム開発者の指定によりハードウェア環境を切り換えるながら、被デバッグプログラムのデバッグを行うことができるので、各々のハードウェア環境の欠点を他のハードウェア環境の優位点が補う形でプログラムを実行することができ、効率のよいプログラムのデバッグを行なうことができる。

【0141】例えはあるハードウェア環境がシミュレー

タにより構成されるものであり、解析機能を豊富に持ちながらも試作機のハードウェア環境を利用しているハードウェア環境と比較して膨大な実行時間を要する場合は、途中までの被デバッグプログラムの実行を試作機のハードウェア環境に行わせ、肝心のバグ原因解析をシミュレータの豊富な機能を用いて行わせることができる。

【0142】数十回、数百回の被デバッグプログラム実行につき一回発生するという発生頻度が非常に低いバグの原因追究時に、シミュレータの解析機能を用いたい場合、本来ならシミュレータにより被デバッグプログラムの実行を数十回、数百回繰り返させる必要がある。しかし上記のようなハードウェア環境の切り換えを利用すれば、試作機のハードウェア環境に数十回、数百回の被デバッグプログラムの繰り返しを行わせ、シミュレータをバグ原因の追究のみに用いることができる。このようにハードウェア環境の切り換えを応用すれば、数十回、数百回の被デバッグプログラムの実行が早く行え、尚且つ豊富な解析機能が利用できるのでデバッグ作業に要する時間が大きく短縮される。

【0143】更にこの切り換えは、コマンド発行という手軽な操作であるので、操作者はデバッグ中のどのような状況においても、思い通りに行える。そのためバグが発生した状況を他のデバッグ装置で再現する手間に煩わされることなく、別の実行環境にてそのバグの発生原因を追究することができ、貴重な開発時間を有効に活用することができる。

【0144】またその切り換えは、バグ発生当時の保持値を他のハードウェア環境に設定し、マイクロコンピュータやマイクロプロセッサのプログラムを最初から実行し直すという煩わしさを伴わずに行われるので、貴重な開発時間を有効に活用できる。また、前記ターゲットマシンは、動作状態情報の入出力を第1の様式にて行い、前記エミュレータマシンは、動作状態情報の入出力を第2の様式にて行い、前記ソフトウェアシミュレータは、動作状態情報の入出力を第3の様式にて行い、前記変換手段は、切換元がターゲットマシンであり切換先がソフトウェアシミュレータであれば、第1の様式を有する動作状態情報を第3の様式に変換する第1変換部と、切換元がエミュレータマシンであり切換先がソフトウェアシミュレータであれば、第2の様式を有する動作状態情報を第3の様式に変換する第2変換部とを備え、前記設定手段は、動作状態情報が第3の様式に変換されれば、動作状態情報をソフトウェアシミュレータに設定する第1設定部を備えるように構成してもよい。この構成では、たとえ動作状態情報の入出力が3つの様式を用いて行われていても、ターゲットマシン、エミュレータマシンは動作状態情報を相互にやりとりすることができる。

【0145】また、上記のように構成されたデバッグ装置における前記変換手段は、切換元がソフトウェアシミュレータであり切換先がターゲットマシンであれば、第

3の様式を有する動作状態情報を第1の様式に変換する第3変換部と、切換元がソフトウェアシミュレータであり切換先がエミュレータマシンであれば、第3の様式を有する動作状態情報を第2の様式に変換する第4変換部とを備え、前記設定手段は、動作状態情報が第1の様式に変換されれば、動作状態情報をターゲットマシンに設定する第2設定部と、動作状態情報が第2の様式に変換されれば、動作状態情報をエミュレータマシンに設定する第3設定部とを備えるように構成してもよい。この構成では、たとえ動作状態情報の入出力が3つの様式を用いて行われていても、ターゲットマシン、エミュレータマシンは動作状態情報を相互にやりとりすることができる。

【0146】また前記変換手段は、切換元がターゲットマシンであり切換先がエミュレータマシンであれば、第1変換部を制御して、第1の様式を有する動作状態情報を第3の様式に変換させ、変換後、第4変換部を制御して、第3の様式を有する動作状態情報を第2の様式に変換させる第1制御部と、切換元がエミュレータマシンであり切換先がターゲットマシンであれば、第2変換部を制御して、第2の様式を有する動作状態情報を第3の様式に変換させ、変換後、第3変換部を制御して、第3の様式を有する動作状態情報を第1の様式に変換させる第2制御部とを備えるように構成してもよい。この構成では、たとえ動作状態情報の入出力が3つの様式を用いて行われていても、ターゲットマシン、エミュレータマシンは動作状態情報を相互にやりとりすることができる。

【0147】また前記エミュレータ内部のプロセッサと第1回線により接続され、第1の様式に変換された動作状態情報の入出力をを行うシリアル転送ポートと、ターゲットマシン内部のインサーキットエミュレータマシンと第2回線により接続され、第2の様式に変換された動作状態情報の入出力をを行うエミュレータマシンドライバ部とを備えるように構成してもよい。

【0148】またターゲットマシン、エミュレータマシン、ソフトウェアシミュレータが有するレジスタ及びメモリを、有効値が格納されているものと、無効値が格納されているものとに分類して管理する記憶資源管理手段と、読み出手段により切換元動作検証対象から動作状態情報が読み出されると、記憶資源管理手段の管理内容を参照して、読み出された動作状態情報に含まれている無効値をデフォルト値に書き換える書換手段とを備え、前記変換手段は、無効値がデフォルト値に書き換えられた動作状態情報を、切換先動作検証対象の様式に変換するように構成してもよい。この構成によれば、プログラム開発者がハードウェア環境を切替る際、動作状態情報に含まれていた無効な値をデフォルト値に書き換えるので、スムーズなプログラムのデバッグを行うことができる。

【0149】加えて、切換先、切換元においてレジスタの実装状態が異なっていても、その差違している部分に

予め定めた値を設定することができる。故に周辺機器レジスタの制御を含む被デバッグプログラムのデバッグも行うことが可能になる。またターゲットマシン、エミュレータマシン、ソフトウェアシミュレータが有するメモリにおいて、ブレークポイントの設定が可能な領域と、不可能な領域とに分割して管理する設定可否領域管理手段と、動作状態情報が変換されると、設定可否領域管理手段を参照して、ブレークポイントが切換先動作検証対象においても同様に設定することができるか否かを判定する判定手段と、切換先動作検証対象においてブレークポイントが設定できない場合、切換先ではブレークポイントが設定できない旨を操作者に通知する通知手段とを上記構成に付加してもよい。

【0150】この構成によれば、これまで切換元側で使用しつつ続けていたブレークポイントのうち一部が切換先において使用不可能となる場合、切替後は該ブレークポイントが使えないことをプログラム開発者に察知させることができる。ここで各動作検証対象が蓄積できるアドレスの最大量を管理する最大量管理手段と、動作状態情報が変換されると、最大量管理手段におけるアドレスの最大量を参照して切換先動作検証対象における命令アドレス群の蓄積量を認識する認識手段と、蓄積量を認識すると、切換先動作検証対象のアドレス蓄積量と、切換先動作検証対象とのアドレス蓄積量の違いを操作者に通知する通知手段とを上記構成に付加してもよい。

【0151】この構成によれば、これまで切換元側で蓄積され続けていたトレースデータのうち一部が切換先において欠落する場合、その欠落部に注意する旨をプログラム開発者に察知させることができる。

【図面の簡単な説明】

【図1】本実施形態におけるデバッグ装置の外観図である。

【図2】本実施形態におけるデバッグ装置のハードウェア構成を示す構成図である。

【図3】モニタ型ハードウェア環境の構成を示す図である。

【図4】ICE型ハードウェア環境の構成を示す図である。

【図5】シミュレータ型ハードウェア環境の構成を示す図である。

【図6】デバッグ装置の階層構造を示す図である。

【図7】アプリケーション層11内部の階層構造を示す図である。

【図8】アプリケーション層11内部の各階層に含まれている構成要素を示す図である。

【図9】本実施形態のデバッグ装置において用いることができるデバッグコマンドの体系図である。

【図10】本実施形態のデバッグ装置において用いることができるselectTARGETコマンドの拡張コマンドを示す図である。

【図11】ハードウェア環境切り換えを規定するテーブルのフォーマットを示す図である。

【図12】ユーザインターフェイス層50によるデバッグ作業時のディスプレイ2の表示例である。

【図13】変換前のcopyMEMコマンド～copyTRACE_DATAコマンドと、変換後の関数呼出コードTCI_get_mem(adr, length, unit, image)～TCI_get_TD(adr, length, unit, image)の対応関係を示す図である。

【図14】図13に示した全ての関数呼出コードがどのように書き換えられたかを示す図である。

【図15】ターゲット内ワークエリア38の内容を読み出す際、モニタ制御層61と試作機30との間でなされる通信プロトコル及び当該通信プロトコルに用いられるパケットを示す説明図である。

【図16】ターゲット内ワークエリア38に内容を書き込む際、モニタ制御層61と試作機30との間でなされる通信プロトコル及び当該通信プロトコルに用いられるパケットを示す説明図である。

【図17】ターゲット内ワークエリア38の内容を読み出す際、モニタ制御層61が行う処理のフローチャートである。

【図18】ターゲット内ワークエリア38に内容を書き込む際、モニタ制御層61が行う処理のフローチャートである。

【図19】ICE型ハードウェア環境の内容を読み出す際、ICE制御層62及びICE用デバイスドライバが行う処理の説明図である。

【図20】ICE型ハードウェア環境に内容を書き込む際、ICE制御層62及びICE用デバイスドライバが行う処理の説明図である。

【図21】モニタ型からシミュレータ型へと状態情報が転送される様子を示す説明図である。

【図22】モニタ型からICE型へと状態情報が転送される様子を示す説明図である。

【図23】ICE型からモニタ型へと状態情報が転送される様子を示す説明図である。

【図24】ICE型からシミュレータ型へと状態情報が転送される様子を示す説明図である。

【図25】シミュレータ型からモニタ型へと状態情報が転送される様子を示す説明図である。

【図26】シミュレータ型からICE型へと状態情報が転送される様子を示す説明図である。

【図27】(a)ブレークポイント設定先アドレスの一例を示す図である。

(b)割り込み信号発生要件の一例を示す図である。

(c)トレースデータの一例を示す図である。

【図28】(a)ブレークポイント管理機能(1)のための管理情報の一例を示す図である。

(b)トレースデータ蓄積サイズ管理機能(2)のための管理情報の一例を示す図である。

(c) 割り込み発生条件管理機能(3)のための管理情報の一例を示す図である。

【図29】I/Oマッピング情報の管理機能(4)のための管理情報の一例を示す図である。

【図30】本実施形態のselectTARGETコマンド実行時のデバッグ装置の処理内容を示すフローチャートである。

【図31】本実施形態のTargetASSIGNコマンド実行時のデバッグ装置の処理内容を示すフローチャートである。

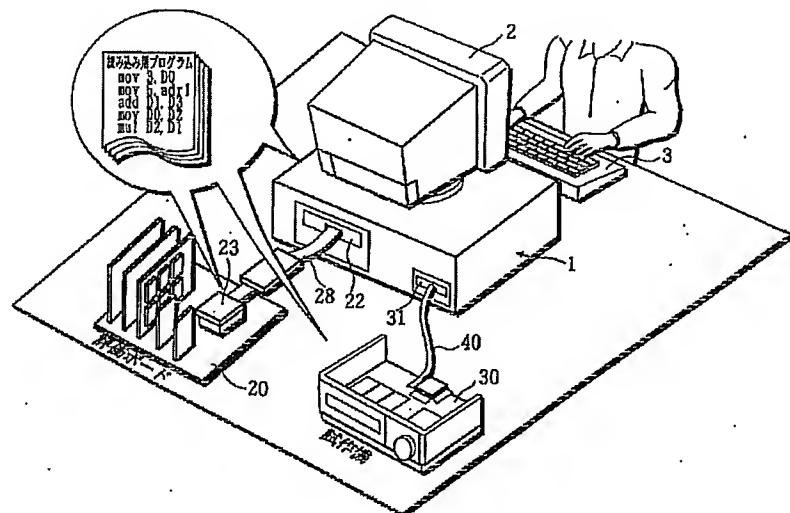
【図32】本実施形態のIFBreakコマンド実行時のデバッグ装置の処理内容を示すフローチャートである。

【図33】本実施形態のSwitch(var)実行時のデバッグ装置の処理内容を示すフローチャートである。

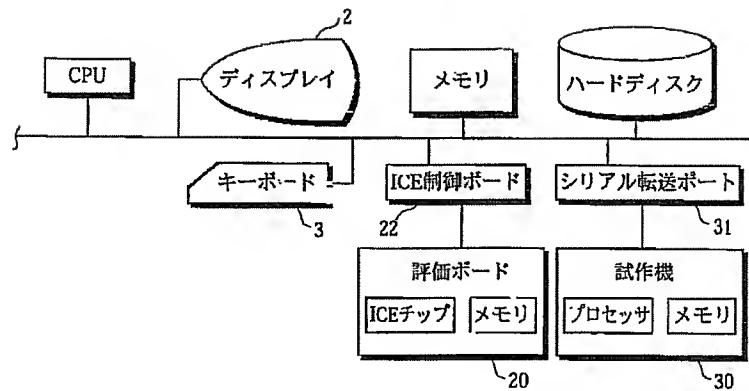
【符号の説明】

1	ホストコンピュータ	27	デバッグ情報エリア
2	ディスプレイ	28	パラレル回線
3	キーボード	30	試作機
10	シェミレータ	31	シリアル転送ポート
11	アプリケーション層	33	プロセッサ
12	オペレーティングシステム層	34	メモリ
13	BIOS層	35	周辺I/Oレジスタ群
14	物理層	36	モニタリングプログラム
20	評価ボード	37	被デバッグプログラム
21	デバイスドライバ	38	ターゲット内ワークエリア
22	制御ボード	39	専用端子
24	メモリ	40	シリアル回線
25	レジスタ群	50	ユーザインターフェイス層
26	スーパバイザプログラム	51	デバッグ核層
		52	ターゲット相互接続層
		53	ターゲット依存層
		61	モニタ制御層
		62	ICE制御層
		72	コマンドインタプリタ
		73	コマンド変換ルーチン
		74	様式変換ルーチン
		75	ワークエリアコンテンツ解析モジュール
		77	共用ワークエリア
		95	プロセッサ模擬ルーチン
		96	メモリ模擬ルーチン
		97	レジスタ模擬ルーチン

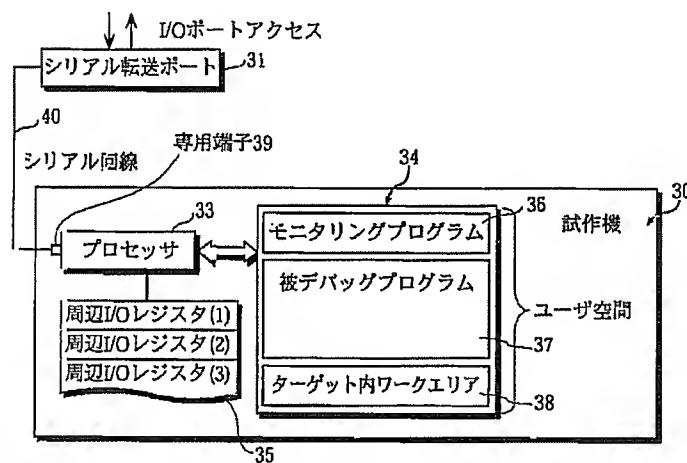
【図1】



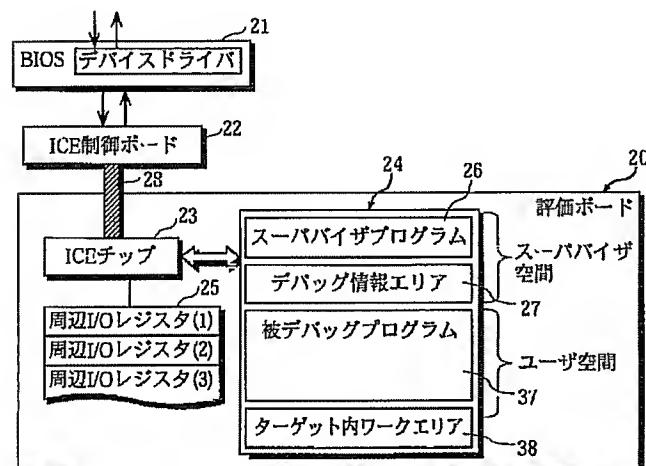
【図2】



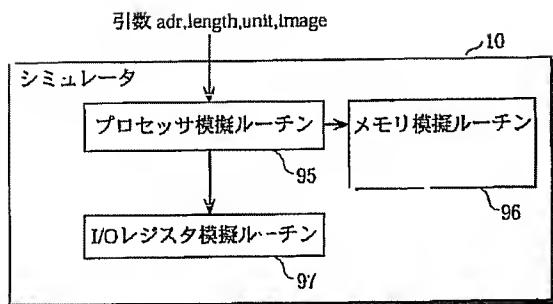
【図3】



【図4】



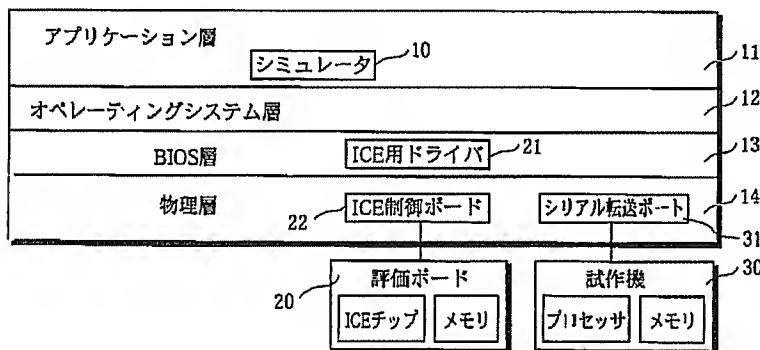
【図5】



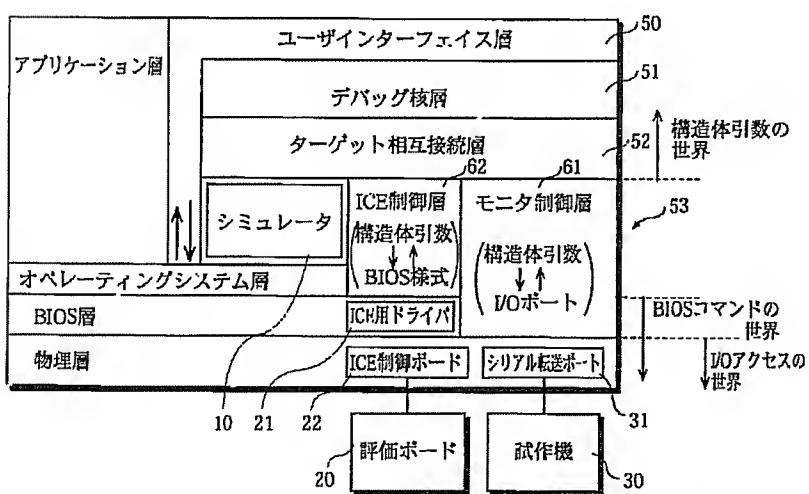
【図11】

先頭アドレス1	最終アドレス1	target 1
先頭アドレス2	最終アドレス2	target 2
先頭アドレス3	最終アドレス3	target 3
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮
先頭アドレスn	最終アドレスn	target n

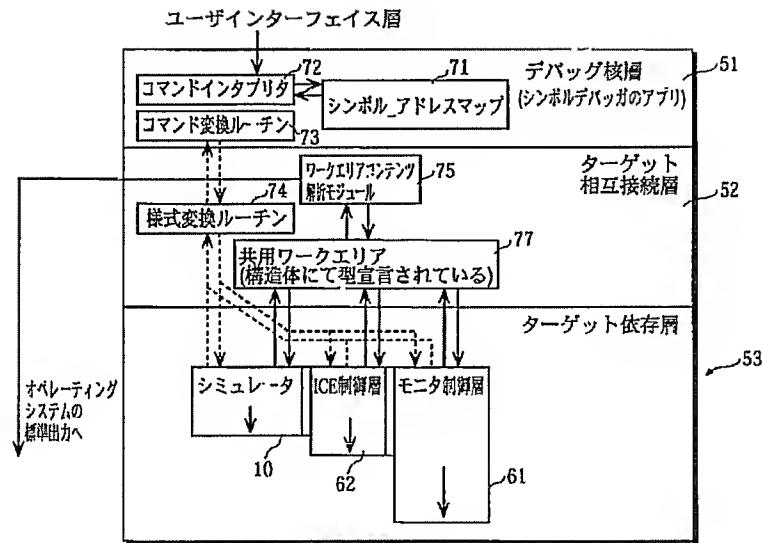
【図6】



【図7】



【図8】



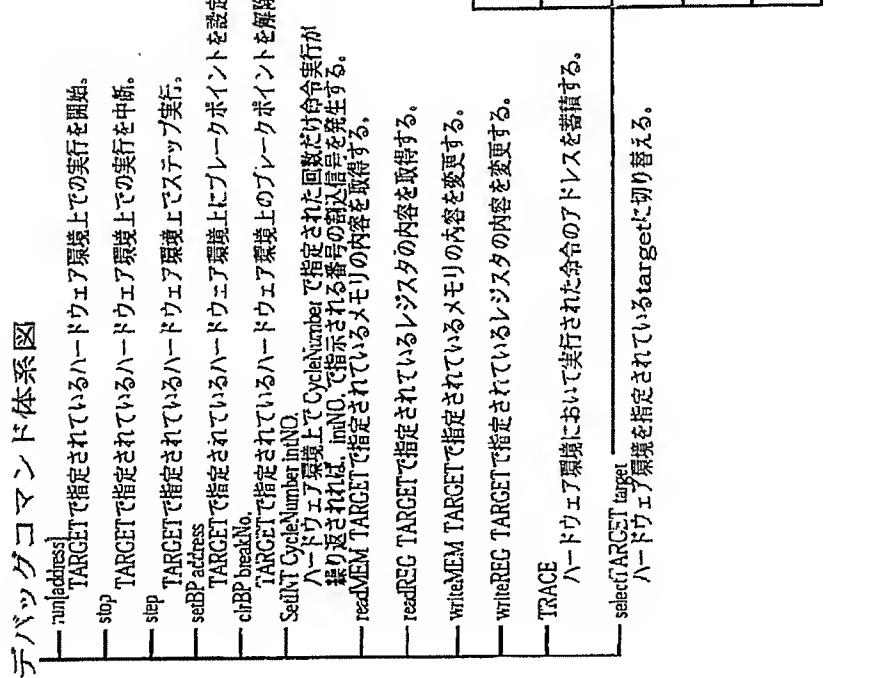
【図10】

IFBreak (BreakNO) {SelectTARGET target}	BreakNOで指定されたブレークポイントにおいて プログラムの動作が停止すればSelectTARGET を実行する。
TargetASSIGN TABLE *	プログラム内の実行領域の割り当てをTABLE * に基づいて行う。
Switch(var[target0]) case'*1': [target 1] case'*2': [target 2]	ハードウェア環境 target 0 の変数 var が'*1'ならば ハードウェア環境 target 1 を起動し、'*2'ならば target 2 を起動する。

【図13】

サブコマンド (変換前)	上位関数の呼出コード (変換後)
CopyMEM	TCI_get_mem(adr,length,unit,image) TCI_set_mem(adr,length,unit,image)
CopyReg	TCI_get_reg(regs,mask) TCI_set_reg(regs,mask)
CopyBP	TCI_get_BP(adr,length,unit,image) TCI_set_BP(adr,length,unit,image)
CopyINT	TCI_get_INT(cycle,intkind) TCI_set_INT(cycle,intkind)
CopyI/O	TCI_get_I/O(adr,length,unit,image) TCI_set_I/O(adr,length,unit,image)
CopyTRACE_DATA	TCI_get_TD(adr,length,unit,image) TCI_set_TD(adr,length,unit,image)

【図9】



【図12】

【図14】

SRC_TARGT = モニタ,DST_TARGET = シミュレータの組み合わせの場合
上位関数の呼出コード(変換前) 下位関数の呼出コード(変換後)

TCI_get_mem(adr,length,unit,image) → MON_get_mem(adr,length,unit,image)
TCI_set_mem(adr,length,unit,image) → SIM_set_mem(adr,length,unit,image)

TCI_get_reg(regs,mask) → MON_get_reg(regs,mask)
TCI_set_reg(regs,mask) → SIM_set_reg(regs,mask)

TCI_get_BP(adr,length,unit,image) → MON_get_BP(adr,length,unit,image)
TCI_set_BP(adr,length,unit,image) → SIM_set_BP(adr,length,unit,image)

TCI_get_INT(cycle,intkind) → MON_get_INT(cycle,intkind)
TCI_set_INT(cycle,intkind) → SIM_set_INT(cycle,intkind)

TCI_get_I/O(adr,length,unit,image) → MON_get_I/O(adr,length,unit,image)
TCI_set_I/O(adr,length,unit,image) → SIM_set_I/O(adr,length,unit,image)

TCI_get_TD(adr,length,unit,image) → MON_get_TD(adr,length,unit,image)
TCI_set_TD(adr,length,unit,image) → SIM_set_TD(adr,length,unit,image)

【図15】

MON_get_mem(addr,length,unit,image)
 により関数呼び出しがされた場合
 シリアル回線を介してモニタプログラムと
 以下の通信シーケンスを実行する

- (1) モニタ制御層→モニタプログラム MEMREAD
 モニタ制御層←モニタプログラム ACK
 モニタ制御層←モニタプログラム DATA1
- モニタ制御層←モニタプログラム DATAN
- (2) 送信パケット(モニタ制御層→モニタプログラム)

0	1	2	3	4	5	6	7	
MEMREAD	CMD/UNIT	SIZE	1	ADRS				

受信パケット(モニタ制御層←モニタプログラム)

0	1	2	3	4	5	6	7	
ACK	ERR?	NULL						
0	1	2	3	4	5	6	7	
DATA1h	DATA							

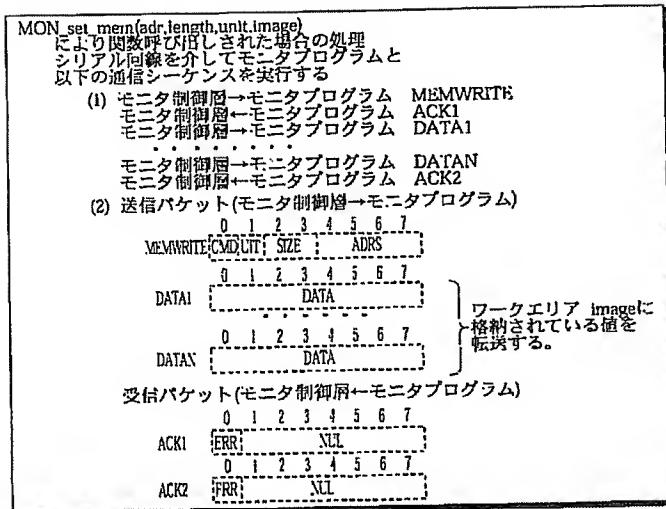
0	1	2	3	4	5	6	7
DATA2h	DATA						

引数数に指定された
 ワイドエリア image1c
 格納する。

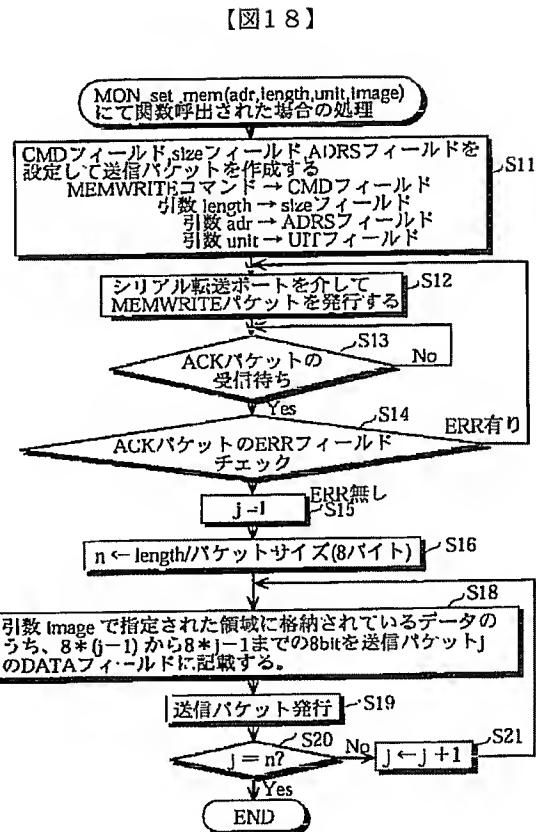
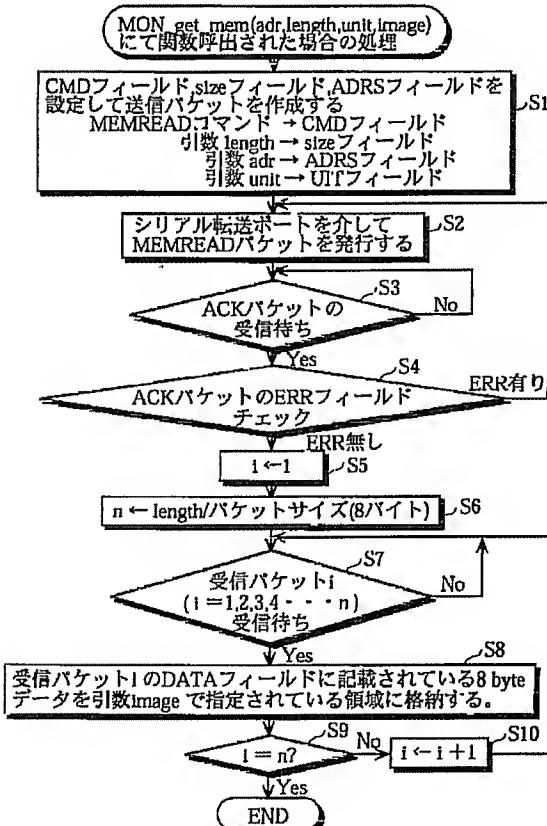
【図29】

target1	I/O_REGISTER_ADDR1	PSW=STATUS1	0x1F
		PSW=STATUS2	0x2A
		PSW=STATUS3	0x3B
	I/O_REGISTER_ADDR2	PSW=STATUS1	0x41
		PSW=STATUS2	0x51
		PSW=STATUS3	0x56
	I/O_REGISTER_ADDR3	PSW=STATUS1	0x18
		PSW=STATUS2	0x22
		PSW=STATUS3	0x24
target2	I/O_REGISTER_ADDR1	PSW=STATUS1	0x33
		PSW=STATUS2	0x32
		PSW=STATUS3	0x41
	I/O_REGISTER_ADDR2	PSW=STATUS1	0x54
		PSW=STATUS2	0x56
		PSW=STATUS3	0x5B
	I/O_REGISTER_ADDR3	PSW=STATUS1	0xA1
		PSW=STATUS2	0x98
		PSW=STATUS3	0x75
target3	I/O_REGISTER_ADDR1	PSW=STATUS1	0x18
		PSW=STATUS2	0x12
		PSW=STATUS3	0x33
	I/O_REGISTER_ADDR2	PSW=STATUS1	0x42
		PSW=STATUS2	0x52
		PSW=STATUS3	0xB2
	I/O_REGISTER_ADDR3	PSW=STATUS1	0x31
		PSW=STATUS2	0x54
		PSW=STATUS3	0x52

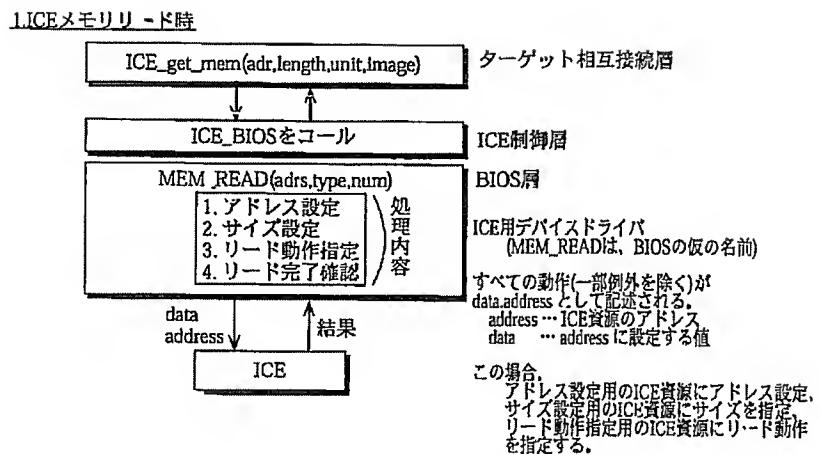
【図16】



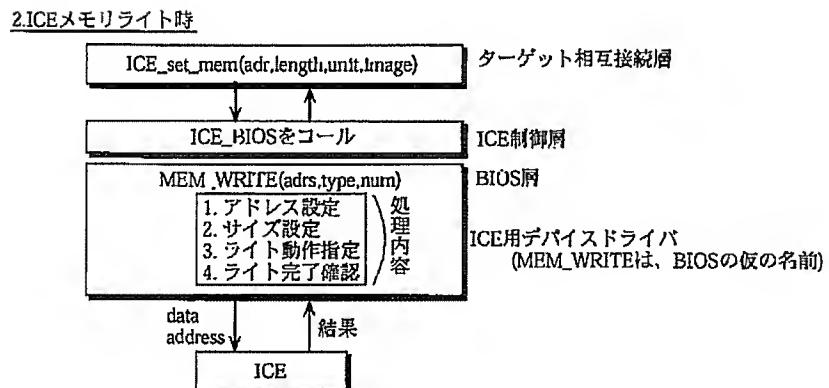
【図17】



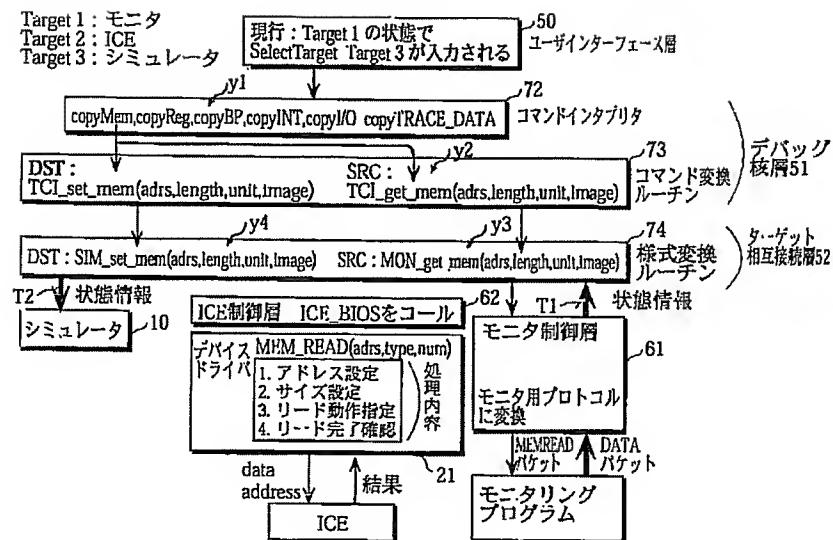
【図19】



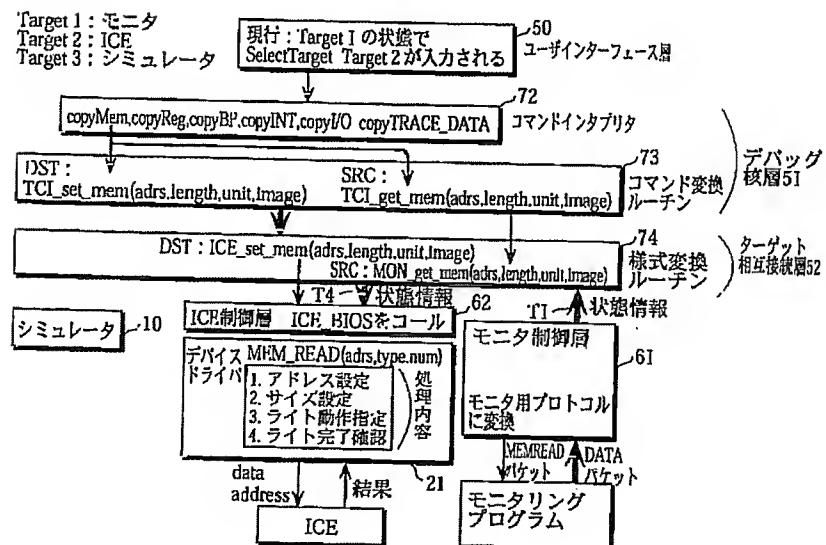
【図20】



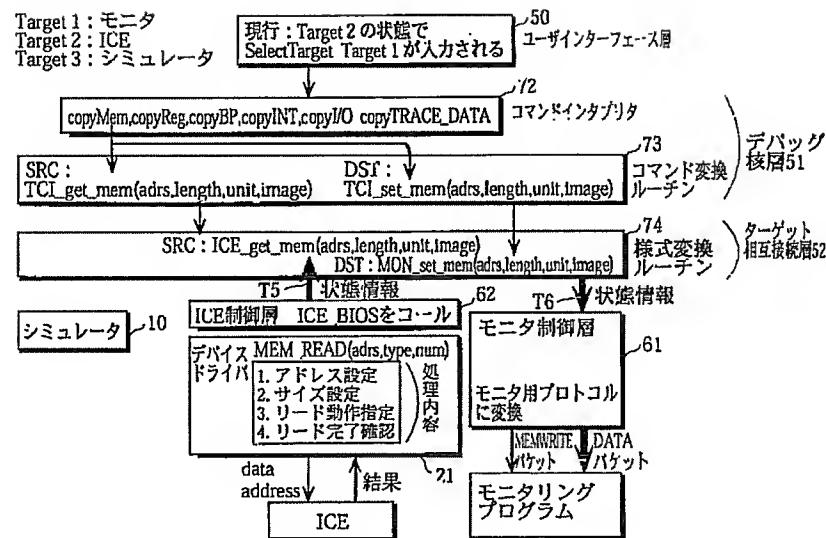
[図21]



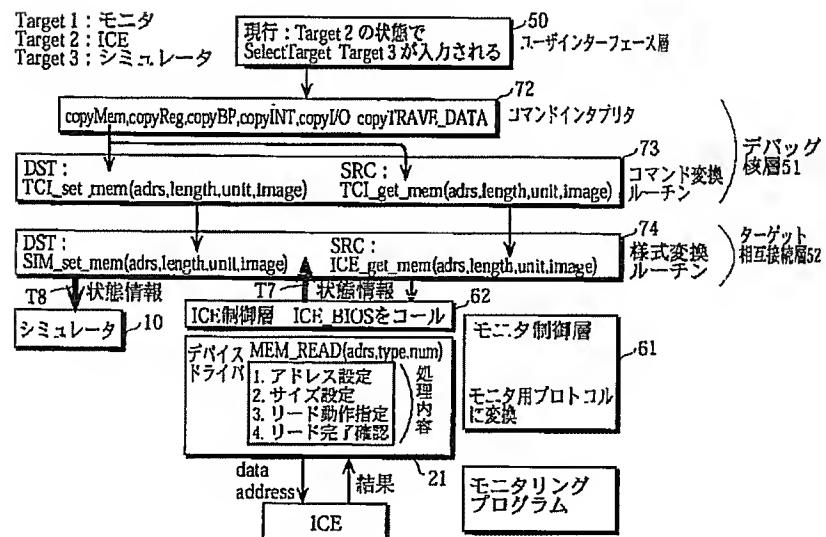
【図22】



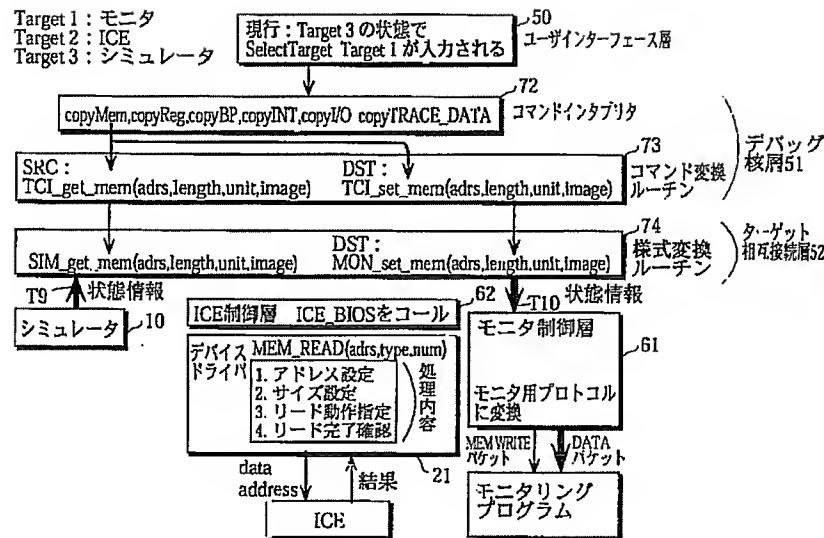
【図23】



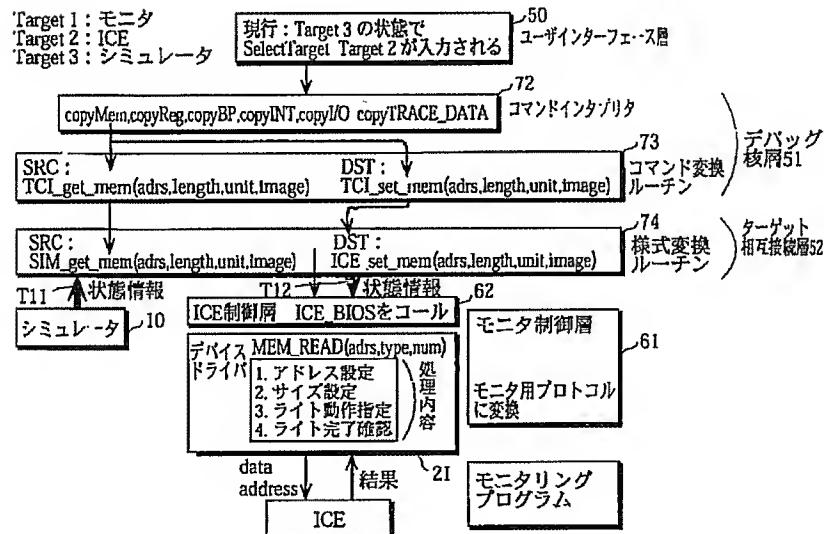
【図24】



【図25】



【図26】



【図27】

(a)

```
/*ブレークポイント設定先アドレス*/
BREAK_POINT[1] = 0x0020
BREAK_POINT[2] = 0x0250
BREAK_POINT[3] = 0x0300
BREAK_POINT[4] = 0x0350
BREAK_POINT[5] = 0x0510
BREAK_POINT[6] = 0x0530
```

(b)

```
/*割り込み信号発生要件*/
CONDITION[1] = 20cycle, Int3
CONDITION[2] = 150cycle, Int5
CONDITION[3] = 230cycle, Int6
CONDITION[4] = 250cycle, Int7
CONDITION[5] = 300cycle, Int8
```

(c)

```
/*トレースデータ*/
0x0000, 0x0001, 0x0002, 0x0003, 0x0004, 0x0005,
0x0006, 0x0007, 0x0008, 0x0009, 0x000A, 0x000B,
0x000C, 0x000D, 0x000E, 0x000F, 0x0010, 0x0011,
0x0012, 0x0013, 0x0014, 0x0015, 0x0016, 0x0017,
0x0018, 0x0019, 0x001A, 0x001B, 0x001C, 0x001D,
0x001E, 0x001F, 0x0020, 0x0021, 0x0022, 0x0023,
```

【図28】

(a)

target1	VALID_AREA1	START_ADDR1 ~ END_ADDR1
	VALID_AREA2	START_ADDR2 ~ END_ADDR2
	VALID_AREA3	START_ADDR3 ~ END_ADDR3
	VALID_AREA4	START_ADDR4 ~ END_ADDR4
target2	VALID_AREA1	START_ADDR1 ~ END_ADDR1
	VALID_AREA3	START_ADDR3 ~ END_ADDR3
	VALID_AREA5	START_ADDR5 ~ END_ADDR5
	VALID_AREA6	START_ADDR6 ~ END_ADDR6
target3	VALID_AREA2	START_ADDR2 ~ END_ADDR2
	VALID_AREA4	START_ADDR4 ~ END_ADDR4
	VALID_AREA5	START_ADDR5 ~ END_ADDR5
	VALID_AREA7	START_ADDR7 ~ END_ADDR7

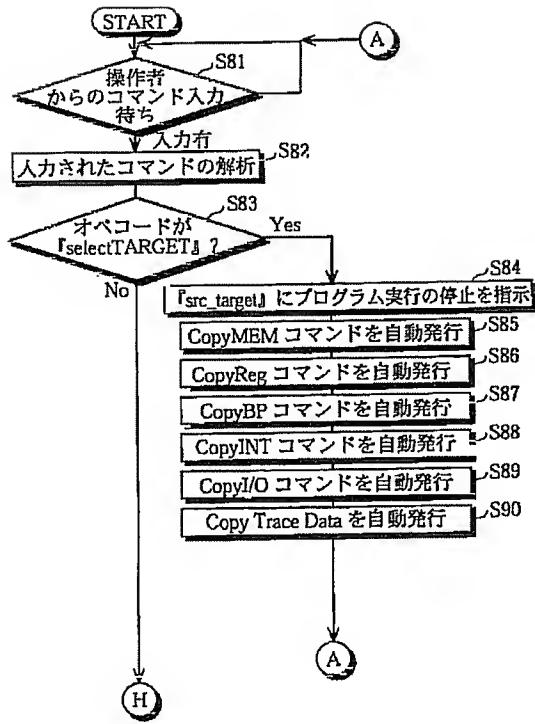
(b)

target1	SPACE_FOR_TRACE_DATA	MAX 16Kbyte
target2	SPACE_FOR_TRACE_DATA	MAX 1Kbyte
target3	SPACE_FOR_TRACE_DATA	MAX 32Kbyte

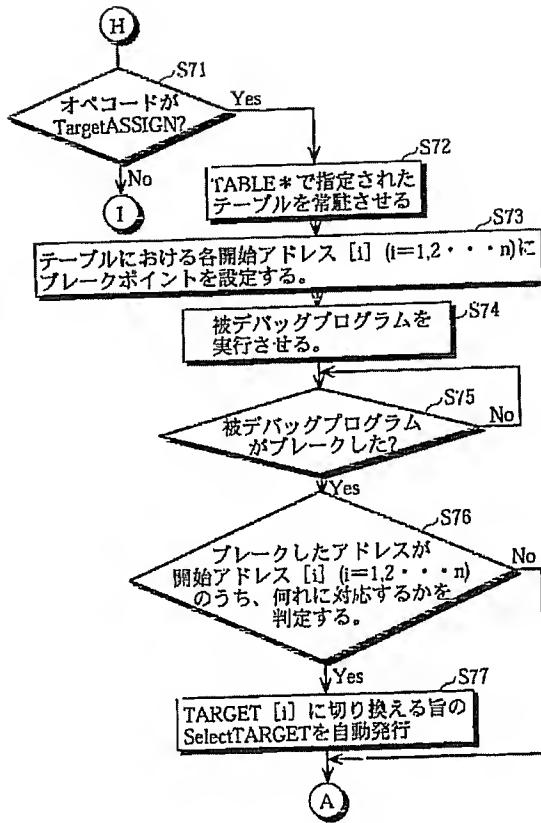
(c)

target1	CYCLE_NUMBER_COUNT	INVALID
target2	CYCLE_NUMBER_COUNT	VALID
target3	CYCLE_NUMBER_COUNT	VALID

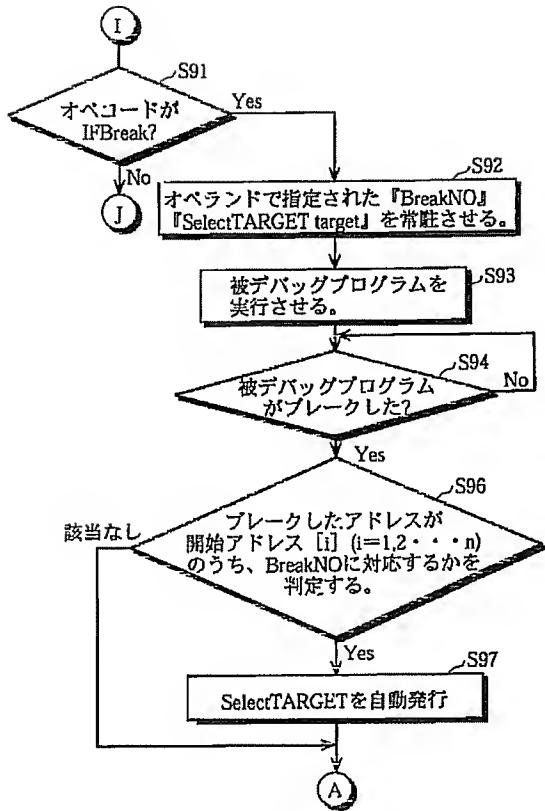
【図30】



【図31】



【図32】



【図33】

